

MULTI: Editing Files and Configuring the IDE



Green Hills Software, Inc.

**30 West Sola Street
Santa Barbara, California 93101
USA**

**Tel: 805-965-6044
Fax: 805-965-6343
www.ghs.com**

DISCLAIMER

GREEN HILLS SOFTWARE, INC. MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. Further, Green Hills Software, Inc. reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation of Green Hills Software, Inc. to notify any person of such revision or changes.

Copyright © 1983-2009 by Green Hills Software, Inc. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior written permission from Green Hills Software, Inc.

Green Hills, the Green Hills logo, CodeBalance, GMART, GSTART, INTEGRITY, MULTI, and Slingshot are registered trademarks of Green Hills Software, Inc. AdaMULTI, Built with INTEGRITY, EventAnalyzer, G-Cover, GHnet, GHnetLite, Green Hills Probe, Integrate, ISIM, u-velOSity, PathAnalyzer, Quick Start, ResourceAnalyzer, Safety Critical Products, SuperTrace Probe, TimeMachine, TotalDeveloper, DoubleCheck, and velOSity are trademarks of Green Hills Software, Inc.

All other company, product, or service names mentioned in this book may be trademarks or service marks of their respective owners.

PubID: 10693
March 2, 2015

Contents

	Preface	ix
	About This Book	x
	The MULTI 5 Document Set	xi
	Conventions Used in the MULTI Document Set	xii
1	Introduction	1
	The MULTI Integrated Development Environment	2
	MULTI Launcher	2
	Editing Tools	2
	Building Tools	2
	Debugging Tools	3
	Administrative Tools	3
	The MULTI Launcher	4
	MULTI Workspaces	5
	Accessing Online Help	6
	Full Online Manuals	6
	Context-Sensitive Help	6
	Command and Configuration Option Help	6
	The Help Viewer Window	7
	Navigating Manuals in the Help Viewer	8
Part I	MULTI Editing Tools	
2	The MULTI Editor	15
	Overview of Editor Features	16
	Starting the MULTI Editor	16
	Starting the Editor from the Launcher	16
	Starting the Editor as a Stand-Alone Program	17
	Starting the Editor from the Project Manager	18
	Starting the Editor from the Debugger	18
	Editor Window Components	19
	The Editor Toolbar	20
	The File and Procedure Fields	21
	The Source Pane Shortcut Menu	21
	The Status Bar	22

Contents

Navigating in Files	23
Using References and Prototypes	23
Using the GoTo Dialog Box	25
Searching in the Editor	28
Incremental Searching	28
Interactive Searching Using the Search Dialog Box	29
Searching in Files	33
Working with Code	36
Indenting Code	36
Working with Comments	38
Highlighting the Boundaries of the Current Block of Code	38
Working with Columns	39
Working with Multiple Files	40
Merging Files	40
Comparing Files	45
Configuring the Editor	46
Customizing the Editor GUI	46
Configuring the Editor for a Programming Language	47
Configuring Auto-Complete	52
3 Editor GUI Reference	55
The File Menu	56
The Edit Menu	59
The View Menu	62
The Per File Settings Dialog Box	64
The Block Menu	66
The Tools Menu	68
The Version Menu	71
The Config Menu	73
The Windows Menu	74
The Help Menu	75
UNIX Dialog Boxes	76
The File Chooser Dialog Box (UNIX)	76
The Print Setup Dialog Box	77

Contents

Part II Managing the MULTI IDE

4	Launcher GUI Reference	83
	Starting the Launcher	84
	The Launcher Menus	85
	The File Menu	86
	The Components Menu	87
	The Edit Menu	88
	The Utilities Menu	90
	The Config Menu	91
	The Windows Menu	92
	The Help Menu	93
	The Launcher Toolbar	94
	The Detail Pane	97
5	Using MULTI Workspaces and Shortcuts	99
	Overview of Workspaces and Shortcuts	100
	MULTI Workspaces	102
	Creating Workspaces	102
	Changing Workspace Properties and Global Settings	105
	Saving Workspaces	106
	Importing and Exporting Workspaces	106
	Opening Workspaces	107
	Action Sequences and Actions	107
	Creating or Modifying an Action Sequence	107
	Creating or Modifying an Action	109
	Running Actions and Action Sequences	112
	Managing Running Actions	112
	MULTI Shortcuts	113
	Creating MULTI Shortcuts	113
	Saving Shortcuts	114
	Importing and Exporting Shortcuts	114
	Using Variables in the MULTI Launcher	115
	Overview of Variables	115
	Variable Types	116
	Creating or Modifying Variables	117
	Referencing Variables	118

Contents

	Variable Substitution	118
6	Using Version Control with MULTI	121
	Configuring Version Control in MULTI	122
	Using Version Control with the Editor and Project Manager	124
	Version Control Options from the Editor	124
	Version Control Options from the Project Manager	126
	Automatic Checkout	126
	Show Last Edit	127
	Revert to a Previous Version of a File	128
	Integrating with Third-Party Version Control Systems	128
	Integrating with ClearCase	128
	Integrating with CVS	129
	Integrating with PVCS	130
	Integrating with RCS	131
	Integrating with SourceSafe	132
	Integrating with Subversion	134
	Troubleshooting	135
	Using Subversion or CVS in a Cygwin Environment (Windows)	135
7	Version Control Tools	137
	The Checkout Browser	138
	Starting the Checkout Browser	139
	Using the Checkout Browser	139
	The Checkout Browser Toolbar	143
	The Checkout Browser Menus	145
	The Status Bar and Status Window	148
	Features and Issues Specific to CVS	149
	Using the CVS Checkout Editor	149
	The Commit Changes Dialog Box	152
	Commit Changes Options	153
	The History Browser	154
	Using the History Browser	154
	The History Browser File Menu	156
	The History Browser Shortcut Menu	156
	The Diff Viewer	157
	Opening the Diff Viewer from the Command Line	158

Contents

Using the Diff Viewer	160
Part III Configuring the MULTI IDE	
8 Configuring and Customizing MULTI	169
Setting Configuration Options	170
Using the Options Window	170
Using the configure Command	171
Saving Configuration Settings	172
Propagating Configuration Settings	174
Creating and Editing Configuration Files	175
Loading Configuration Files	178
Clearing Configuration Settings	179
Creating Custom Functionality Using Scripts and Macros	179
Using Script Files	180
Customizing the GUI	181
Configuring and Customizing Toolbar Buttons	183
Creating and Working with Icons	185
Configuring and Customizing Menus	186
Opening and Accessing Menus	190
Customizing Keys and Mouse Behavior	191
Customizing Keys with the keybind Command	191
Customizing Mouse Behavior with the mouse Command	193
Key and Mouse Locations	195
Key and Mouse Command Special Sequences	196
Inserting a Character Blocked By a Custom Key Binding	197
Configuring Taskbar Organization	197
Taskbar Organizer Menu Options	198
Configuring Window Docking	199
UNIX Docking Limitations	200
Configuring Window Focus	201
Configuring File Extensions	202
Extension Mapping Files	202
Configuring File Associations (Windows only)	203

Contents

9	Configuration Options	205
	General Configuration Options	208
	The General Options Tab	208
	Other General Configuration Options	221
	Project Manager Configuration Options	226
	Debugger Configuration Options	230
	The Debugger Options Tab	230
	Other Debugger Configuration Options	241
	MULTI Editor Configuration Options	251
	The MULTI Editor Options Tab	251
	Other MULTI Editor Options	259
	Session Configuration Options	260
	Colors Configuration Options	265
	The More Color Options Dialog Box	269
	The Color Chooser	269
	Deprecated Configuration Options	270

Part IV Appendices

A	Editor Commands	273
	Auto-Completion Commands	275
	Block Commands	277
	Clipboard Commands	279
	Configuration Commands	282
	Drag-and-Drop Commands	283
	File Commands	285
	Help Commands	289
	if Conditional Commands	290
	Indentation Commands	292
	Insert Commands	293
	Mode Commands	295
	Navigation Commands	296
	Search Commands	301
	Selection Commands	302
	Tag Commands	307

Contents

	Text Deletion Commands	309
	Tools Commands	310
	Undo/Redo Commands	312
	Version Control Commands	313
	View Commands	315
	Deprecated Commands	317
B	Third-Party Tools	319
	Third-Party Version Control Systems	320
	Third-Party Editors	320
	Using the Editor with Third-Party Tools	321
C	Default Key and Mouse Bindings	323
	Default Keyboard Shortcuts	324
	Navigating	324
	Opening, Saving, and Closing	327
	Undo/Redo	328
	Cutting, Copying, and Pasting	328
	Deleting Text	330
	Selecting Text	331
	Searching	332
	Auto-Completion	333
	Indenting Text	334
	Version Control	335
	Miscellaneous	335
	Default Mouse Bindings	336
	First (Left) Mouse Button	336
	Second (Middle) Mouse Button	337
	Third (Right) Mouse Button	338
	Index	339

Contents

Preface

This Preface Contains:

- About This Book
- The MULTI 5 Document Set
- Conventions Used in the MULTI Document Set

This section discusses the purpose of the manual, typographical conventions used, and the MULTI documentation set.

About This Book

The *MULTI: Editing Files and Configuring the IDE* book contains four parts:

- *Part I: MULTI Editing Tools* contains information about using the MULTI Editor. This part also includes descriptions of all Editor menus and the equivalent commands, and provides instructions about configuring the Editor for your programming language. See page 13.
- *Part II: Managing the MULTI IDE* contains information about using the MULTI Launcher, and configuring and using version control to track changes to your files. It also provides information about the following version control tools included with MULTI: the **History Browser**, **Checkout Browser**, and **Diff Viewer**. See page 81.
- *Part III: Configuring the MULTI IDE* contains information about configuring the Integrated Development Environment to help you work more efficiently. You can customize the appearance and functionality of many GUI buttons, menus, keyboard shortcuts and mouse bindings. See page 167.
- *Part IV: Appendices* contains additional reference material, including comprehensive documentation of Editor commands, an explanation of how to use third-party tools with the Editor, and a listing of default key and mouse bindings. See page 271.



Note New or updated information may have become available while this book was in production. For additional material that was not available at press time, or for revisions that may have become necessary since this book was printed, please check your MULTI installation directory for release notes, **README** files, and other supplementary documentation.

The MULTI 5 Document Set

The primary documentation for using MULTI is provided in the following books:

- *MULTI: Getting Started* — Describes how to install MULTI, and takes you through creating, building, and debugging an example project.
- *MULTI: Licensing* — Describes how to obtain, install, and administer Green Hills licenses. This book is intended for system administrators.
- *MULTI: Editing Files and Configuring the IDE* — Describes how to use the MULTI Editor and MULTI Launcher, how to use a version control system with MULTI, and how to configure the MULTI IDE.
- *MULTI: Building Applications* — Describes how to use the MULTI Project Manager and compiler drivers and the tools that compile, assemble, and link your code. Also describes the Green Hills implementation of supported high-level languages.
- *MULTI: Configuring Connections* — Describes how to set up your target debugging interface for use with MULTI and how to configure connections to your target.
- *MULTI: Debugging* — Describes how to use the MULTI Debugger and its associated tools.
- *MULTI: Debugging Command Reference* — Describes how to use Debugger commands and provides a comprehensive reference of Debugger commands.

For a comprehensive list of the books provided with your MULTI installation, see the **toc_target.pdf** file located in the **manuals** subdirectory of your installation.

Most books are available in the following formats:

- A printed book (select books are not available in print).
- Online help, accessible from most MULTI windows via the **Help** → **Manuals** menu.
- An electronic PDF, available in the **manuals** subdirectory of your installation.

Conventions Used in the MULTI Document Set

All Green Hills documentation assumes that you have a working knowledge of your host operating system and its conventions, including its command line and graphical user interface (GUI) modes. For example, you should know how to use basic commands, how to open, save, and close files, and how to use a mouse and standard menus.

Green Hills documentation uses a variety of notational conventions to present information and describe procedures. These conventions are described below.

Convention	Meaning	Example
bold type	Indicates a: • Filename or pathname • Command • Option • Window title • Menu name or menu choice • Field name • Button name	• C:\MyProjects • setup command • -G option • The Breakpoints window • The File menu • Working Directory: • The Browse button
<i>italic type</i>	Indicates: • Replaceable text • A new term • A book title	• -o <i>filename</i> • A task may be called a <i>process</i> or a <i>thread</i> • <i>MULTI: Debugging</i>

Convention	Meaning	Example
monospace type	Indicates: • Text you should enter as presented • A word or words used in a command or example • Source code • Input/output • A function	• Type <code>help</code> <code>command_name</code> • The wait [-global] command blocks command processing, where -global blocks command processing for all MULTI processes. • <code>int a = 3;</code> • <code>> print Test</code> <code>Test</code> • <code>GHS_System()</code>
ellipsis (...) (in command line instructions)	Indicates that the preceding argument or option can be repeated zero or more times.	debugbutton [name]...
greater than sign (>)	Represents a prompt. Your actual prompt may be a different symbol or string. The > prompt helps to distinguish input from output in examples of screen displays.	<code>> print Test</code> <code>Test</code>
pipe () (in command line instructions)	Indicates that one (and only one) of the parameters or options separated by the pipe or pipes should be specified.	call <i>proc</i> <i>expr</i>
square brackets ([]) (in command line instructions)	Indicate optional arguments, commands, options, and so on. You can either include or omit the enclosed elements. The square brackets should not appear in your actual command.	<code>.macro</code> <i>name</i> [<i>list</i>]

The following command description demonstrates the use of some of these typographical conventions.

gxyz [-option]... *filename*

The formatting of this command indicates that:

- The command **gxyz** should be entered as shown.
- The option *-option* should either be replaced with one or more appropriate options or be omitted.
- The word *filename* should be replaced with the actual filename of an appropriate file.

The square brackets and the ellipsis should not appear in the actual command you enter.

Introduction

Chapter 1

This Chapter Contains:

- The MULTI Integrated Development Environment
- The MULTI Launcher
- Accessing Online Help

This chapter provides an overview of the MULTI Integrated Development Environment.

The MULTI Integrated Development Environment

MULTI is a complete Integrated Development Environment (IDE) designed especially for embedded systems engineers to assist them in compiling, optimizing, debugging, and editing embedded applications.

The MULTI IDE includes the following tools for each part of the software development process. Many of the MULTI tools can be launched from within the IDE or as separate stand-alone programs. Parenthetical text indicates corresponding executable files, which are located in your MULTI installation.

MULTI Launcher

MULTI Launcher (mstart) — The gateway to the MULTI IDE, which allows you to quickly launch any of the primary MULTI tools, access open windows, and manage MULTI workspaces.

Editing Tools

- **MULTI Editor (me)** — A graphical editor for modifying text files.
- **Checkout Browser (mcobrowse)** — A graphical viewer for files managed under a version control system.
- **Diff Viewer (diffview)** — A graphical viewer that displays differences between two text files.

Building Tools

- **MULTI Project Manager (mprojmgr)** — A graphical interface for managing and building projects.
- **Integrate (integrate)** — A graphical utility for configuring tasks, connections, and kernel objects across multiple address spaces when using the INTEGRITY RTOS.

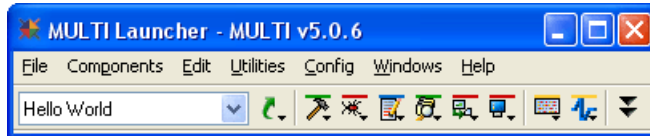
Debugging Tools

- **MULTI Debugger (multi)** — A graphical source-level debugger.
- **Instruction Set Simulator** — A tool that simulates your embedded processor on your development host.
- **EventAnalyzer (mevgui)** — A graphical viewer for monitoring the complex real-time interactions of an embedded RTOS such as INTEGRITY or u-velOSity.
- **ResourceAnalyzer (wperf)** — A graphical viewer for monitoring the CPU and memory usage of an embedded system running the INTEGRITY RTOS.
- **Serial Terminal (mterminal)** — A serial terminal emulator for connecting to serial ports on embedded devices.
- **DoubleCheck** — A static source code analysis tool for locating programming problems that lead to security and reliability failures in software.
- **MULTI TimeMachine Tool Suite** — A wide variety of trace analysis tools that dramatically extend MULTI's trace and debugging capabilities.

Administrative Tools

- **Bug Report (gbugrpt)** — A utility for providing system configuration and tool version information to the Green Hills support staff.
- **Green Hills Probe Administrator (gpadmin)** — A graphical interface for configuring and managing Green Hills Probes and SuperTrace Probes.
- **Graphical Utilities (wgutils)** — A collection of utilities for analyzing and performing various operations on object files, libraries, and executables produced with the Green Hills toolchain.
- **MULTI License Administrator (mlmadmin)** — A graphical utility for managing Green Hills tools licenses.

The MULTI Launcher



The MULTI Launcher (**mstart**) provides a convenient way to launch frequently used tools, to create new or access recently used files and projects, and to manage MULTI workspaces. All of the main MULTI components can be accessed using the following buttons:

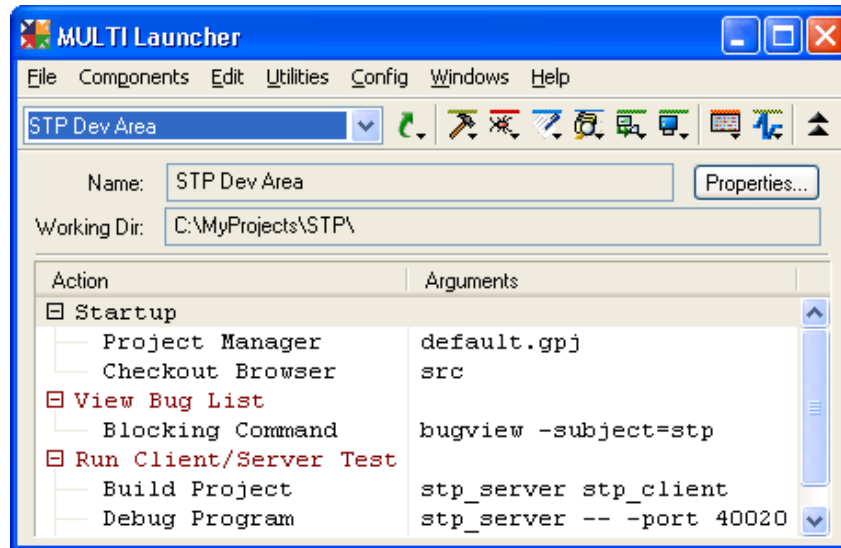
-  — Runs a shortcut or an action sequence in the current workspace. Also allows you to create a new workspace or create or edit a shortcut.
-  — Opens the Project Manager on a recent or new project.
-  — Opens the Debugger on a recent or new executable.
-  — Opens the Editor on a recent or new file.
-  — Opens the Checkout Browser on a recent or new checkout.
-  — Establishes a target connection or opens the Connection Chooser or Connection Organizer.
-  — Opens a Serial Terminal using a recent or new connection.
-  — Opens the EventAnalyzer (licensed separately).
-  — Opens the ResourceAnalyzer (licensed separately).
-  — Shows/hides the detail pane of the Launcher.

You can also launch the Green Hills License Administrator and (if installed) the Green Hills Probe Administrator from the **Utilities** menu.

During development, you can use the MULTI Launcher as a convenient, centralized window manager. You can access any of your open MULTI windows from the **Windows** menu of the Launcher.

MULTI Workspaces

The MULTI Launcher allows you to create and use *workspaces*. A MULTI workspace is a virtual area where the tools, files, and actions required for a particular project can be organized, accessed, and executed.



A workspace is typically created for each Top Project, and includes a working directory and a group of related *actions*—for example, opening a project in the MULTI Project Manager, connecting to a target, or performing a shell command. Actions are grouped into action sequences, so that a single mouse click can perform all the actions in the specified action sequence.

For more information, see Chapter 5, “Using MULTI Workspaces and Shortcuts”.

Accessing Online Help

Online help provides useful information about your MULTI tools and can be accessed in several different ways. The following sections describe how to access online help and how to use the Help Viewer.

Full Online Manuals

You can access an indexed, hypertext version of any MULTI manual by selecting it from the list that appears in the **Help** → **Manuals** menu. The MULTI Help Viewer opens on the manual specified.

You can also access manuals from the command line. Ensure that the MULTI installation directory is in your path and enter the following command:

```
helpview pathname | -m manual_name
```

where:

- *pathname* opens the **.chm** file specified in *pathname*. You must provide a full path.
- -m *manual_name* opens the manual *manual_name*. An example manual name is "MULTI: Debugging". If the manual name contains a space as in the preceding example, you must enclose it with quotation marks.

Context-Sensitive Help

Many MULTI windows and dialog boxes are linked to specific sections of the online manuals. To view the Help Viewer page that documents an active window or dialog box, press F1.

If you are using the MULTI Editor, you can also open context-sensitive help about a button or a menu item by selecting **Help** → **Identify** and then clicking the button or selecting the menu item.

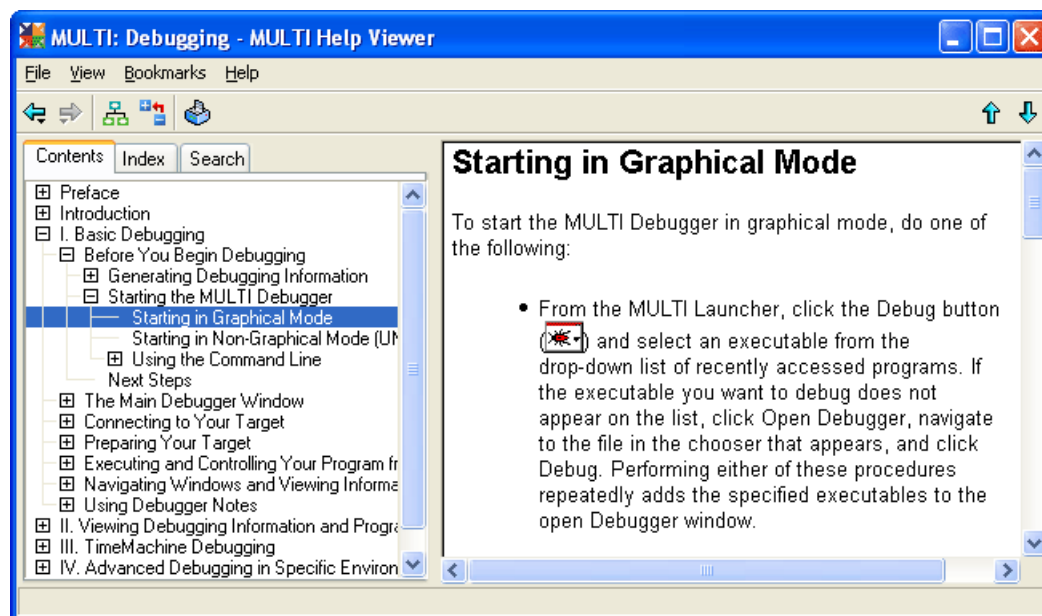
Command and Configuration Option Help

You can obtain help information about a specific MULTI Debugger command or MULTI configuration option by typing `help command_name` or `help configuration_option` in the Debugger command pane. This opens the Help

Viewer on documentation for the specified command or configuration option. You can also type `usage command_name` to print the basic syntax of a MULTI command to the command pane.

The Help Viewer Window

The following graphic displays the Help Viewer window.



The Help Viewer toolbar contains the following buttons, from left to right:

- **Back** (←) and **Forward** (→) — Returns to pages you have visited during the current help session. Click the button once for each page you want to revisit. Click and hold the button to select from a list of pages you have visited. These buttons function across books.
- **View All Subtrees** (📁) — Toggles highlighting of the current selection's sub-sections (if any) in the **Contents** tab, and toggles the display of help for the current selection's sub-sections (if any) in the view pane. To disable highlighting in the **Contents** tab and to display help for only the selected item, click this button again. When you open the Help Viewer, this button defaults to its last value.
- **Contract All Unselected** (📁) — Collapses hierarchies located in the **Contents** tab if the hierarchies do not contain any selections.

- **Print** () — Prints the help page displayed in the view pane. To create a help page consisting of mixed topics, hold down the Ctrl key while clicking separate topics. MULTI combines the separate topics and orders them by their location in the **Contents** tab.
- **Previous Page** () and **Next Page** () (located at far right) — Displays the previous or following section as ordered in the **Contents** tab. If the previous or next section includes sub-sections and the **View All Subtrees** button is enabled, the Help Viewer displays the sub-sections (if any) along with the section.

The MULTI Help Viewer contains two panes. The right-hand pane, or the *view pane*, displays the help page for your selection. The left-hand pane contains the following three tabs:

- **Contents** — Displays the parts, chapters, sections, and sub-sections for the manual you are viewing. Click a plus or minus icon to show or hide headings for embedded sections. Click a heading to display the associated help page in the view pane.
- **Index** — Displays index entries for the manual. To search the index entries, enter a word or words in the text box located at the top of the pane. MULTI returns results that contain the word(s) you typed. Click an index entry to display the corresponding help page in the view pane.
- **Search** — Provides an interface that allows you to search for specific words in the current manual or in all manuals.

Navigating Manuals in the Help Viewer

Navigating manuals in the MULTI Help Viewer is easy. This section describes different methods of finding information.

Viewing Previous and Next Sections

To display the previous or following section as ordered under the **Contents** tab, do one of the following:

- Click the  or  button located in the upper-right corner of the Help Viewer window.
- Select **View** → **Previous Contents Entry** or **View** → **Next Contents Entry**.

If the previous or next section includes sub-sections and the **View All Subtrees** button () is enabled, the Help Viewer displays the sub-sections (if any) along with the section.

Returning to Previously Viewed Help Pages

To return to pages you have visited during the current help session, do one of the following:

- Click the  or  button located in the left corner of the toolbar. Click the button once for each page you want to revisit. Click and hold the button to select from a complete list of pages you have visited during the current help session. These buttons function across books.
- Select **View** → **Back** or **View** → **Forward** from the menu bar. This feature functions across books.

Linking to Related Topics

The underlined links available in the view pane allow you to jump to pages related to the topic you are viewing.

If, when you click a link, a dialog box appears with the message:

You have requested information that the Help Viewer is unable to retrieve.

one of the following factors may be the cause:

- The manual that the link points to may not be included with your distribution. For a list of available manuals, look in the manuals directory (or directories) of your Green Hills Software distribution(s).
- If the link points to an INTEGRITY or u-velOSity manual, you may be using a version of the operating system that does not support links from MULTI help.
- If the link points to an INTEGRITY or u-velOSity manual, you may need to set the location of the installed OS distribution so that the Help Viewer can locate the relevant help files. For information about how to do this, see “Using MULTI with INTEGRITY or u-velOSity” in Chapter 2, “Installation” in the *MULTI: Getting Started* book.



Note If you are using MULTI with multiple target architectures (such as ARM and PowerPC), tools of the same version number should be installed into a single directory, regardless of differing architectures. Note, however, that links to target-specific books may yield unpredictable results in this situation.

Bookmarking Help Pages

You can use bookmarks to return to pages you have visited across MULTI sessions. To bookmark a page, select **Bookmarks** → **Bookmark This Page**.

You can bookmark a help page consisting of mixed topics by holding down the Ctrl key while clicking separate topics. MULTI combines the separate topics in the view pane and orders them by their location under the **Contents** tab. Bookmark the page as usual.

The **Bookmarks** menu lists all your bookmarks from across MULTI manuals and sessions. To return to a bookmarked page, do one of the following:

- Select **Bookmarks** and choose a bookmark.
- Select **Bookmarks** → **Manage Bookmarks**, click a bookmark, and click the **Show** button (.
- From any MULTI tool, select **Help** → **Bookmarks** and choose a bookmark.

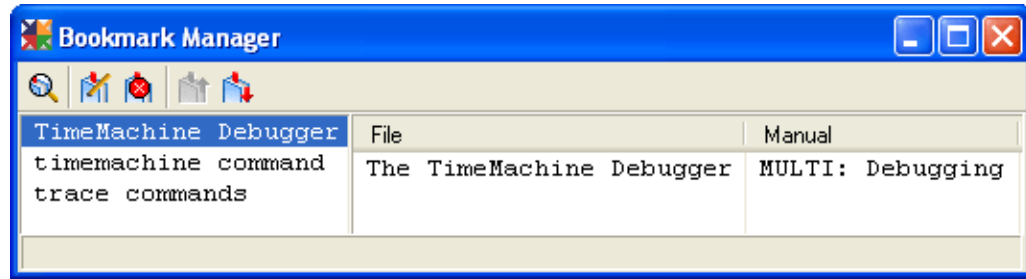
To rename, delete, or reorder existing bookmarks, launch the **Bookmark Manager** as described in the next section, “Managing Bookmarks” on page 10.

Managing Bookmarks

You can use the **Bookmark Manager** to rename and delete bookmarks, navigate to bookmarks, and modify the ordering of bookmarks in the **Bookmarks** menu.

To open the **Bookmark Manager**, do one of the following:

- In the Help Viewer, select **Bookmarks** → **Manage Bookmarks**.
- From any MULTI tool, select **Help** → **Bookmarks** → **Manage Bookmarks**.



The list on the left side of the **Bookmark Manager** displays the names of all the bookmarks that you have created. When a bookmark is selected, the file(s) and the manual marked by that bookmark are displayed in the right side of the window. The bookmark may reference multiple files (help pages) if, for example, you created the bookmark with multiple entries selected in the **Contents** tab or if you selected an index entry that pertained to multiple files.

The toolbar buttons in the **Bookmark Manager** allow you to perform the following operations on bookmarks:

- — Display the selected bookmark(s) in the Help Viewer.
- — Rename the selected bookmark.
- — Delete the selected bookmark(s).
- and — Move the selected bookmark up or down in the list. The order of the bookmarks in the **Bookmarks** menu mirrors the order of the bookmarks in the **Bookmark Manager**.



Note The and buttons are available for use on multiple bookmarks. You can select multiple bookmarks from the left side of the window by dragging to select rows or by using Shift or Ctrl to select items.

Opening Additional Manuals from the Help Viewer

To open a new manual in the current window, do one of the following:

- Select **File** → **Open Manual** and choose from the list of installed manuals.
- Select **File** → **Open Manual File** and navigate to a particular file in the file chooser that appears.

To open a new manual in a new window, select **View** → **Open Other Manuals in New Window** and then open the manual as usual. When you open the Help

Viewer, the **Open Other Manuals in New Window** toggle option defaults to its last value.

MULTI Editing Tools

Part I

The MULTI Editor

Chapter 2

This Chapter Contains:

- Overview of Editor Features
- Starting the MULTI Editor
- Editor Window Components
- Navigating in Files
- Searching in the Editor
- Working with Code
- Working with Multiple Files
- Configuring the Editor

This chapter provides an introduction to the MULTI Editor, which you can use to create and modify text files.

Overview of Editor Features

The Editor provides powerful features to help you work with your files. Details about the following features are provided in this book:

- *Navigating in Files* — The Editor can automatically obtain function prototypes and cross reference information, and provides a number of search options. For more information, see “Navigating in Files” on page 23.
- *Working with Code* — The Editor provides tools and keyboard shortcuts that are helpful when editing source code. For more information, see “Working with Code” on page 36.
- *Working with Multiple Files* — The Editor contains tools to merge or compare multiple files. For more information, see “Working with Multiple Files” on page 40.
- *Configuring the Editor* — The look and functionality of the Editor can be customized to help you work more efficiently. See “Configuring the Editor” on page 46.
- *Version Control* — The Editor is compatible with several version control systems and allows you to perform many version control operations without leaving the Editor. For more information, see “The Version Menu” on page 71.

Starting the MULTI Editor

There are a number of ways to start the Editor, either as a stand-alone editing program, or from other MULTI tools.

Starting the Editor from the Launcher

To start the Editor from the Launcher, do one of the following:

- Click the **Edit** button () and select a file or the **Open Editor** option.
- Select **Components** → **Open Editor**.

Starting the Editor as a Stand-Alone Program

To start the MULTI Editor as a stand-alone program, run the Editor executable from the command line:

me [*options*]

where *options* is any non-conflicting combination of command line options:

+line file
Opens the Editor on <i>file</i> and navigates to line number <i>line</i> .
-diff file1 file2
Opens the Diff Viewer on the specified files.
file
Opens the Editor on <i>file</i> .
-h -usage UNIX only
Displays on-screen usage information.
-help
Launches online help.
-new file1 file2...
Opens each file in a new Editor window.
-reuse file1 file2...
Opens each file in the same Editor window. This is the default.
-showhistory file1 [file2]...
Opens a History Browser on each file specified.

If you do not specify any option, a file chooser appears. Select or navigate to a file to open it in the Editor.



Note The MULTI IDE may reuse an Editor process started from the command line. In this case, the process does not exit until the entire MULTI IDE is closed.

Starting the Editor from the Project Manager

To start the Editor while using the Project Manager, do one of the following:

- Select a file, then click the **Edit** button () to open it in the Editor.
- Select one or more files in the source pane, then select **Edit** → **Edit**, or right-click and select **Edit** from the shortcut menu.
- Double-click a text filename in the source pane.
- Double-click an error or warning in the **Build Details** window to open the source file in the Editor with the cursor placed on the line with the error. To see the next error (if any), click the **Next Error** button ()

Starting the Editor from the Debugger

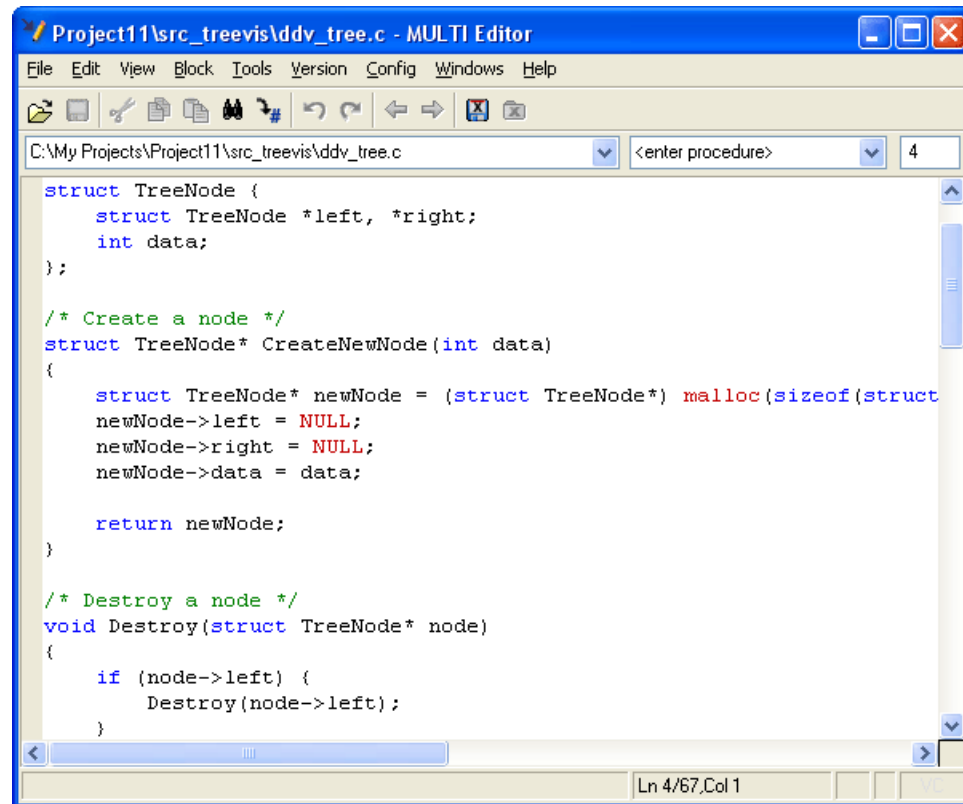
To start the Editor while using the Debugger, do one of the following:

- Click the **Edit** button ()
- Right-click in the source pane and select **Edit File**.
- Enter the **edit** command in the Debugger command pane. For more information about this command and related commands, see “General View Commands” in Chapter 22, “View Command Reference” in the *MULTI: Debugging Command Reference* book.

When you start the Editor from the Debugger, the Debugger creates temporary copies of the source files so you can continue to debug your program using the original source code. Changes that you make in the Editor affect the actual files and not these temporary files, which are deleted when the Debugger or the Editor exits.

Editor Window Components

The following graphic displays the Editor window.



The main components of the Editor window are described below.

- *Title Bar* — Contains the path and filename of the file displayed in the current Editor window.
- *Menu Bar* — Contains menus to access Editor functions. All menus and menu items are described in detail in Chapter 3, “Editor GUI Reference”.
- *Editor Toolbar* — Contains buttons for some of the most common Editor actions. For more information, see “The Editor Toolbar” on page 20.
- *File and Procedure Fields* — Contains fields that provide quick access to files, procedures and line numbers. For more information, see “The File and Procedure Fields” on page 21.
- *Source Pane* — Displays the text of the active document.
- *Status Bar* — Displays information about document status and cursor position. For more information, see “The Status Bar” on page 22.

The Editor Toolbar

The Editor Toolbar contains the following buttons that provide access to some of the most common Editor actions:

-  **(Open)** — Opens the **Edit File** dialog box, which is used to select a file to open in the Editor.
-  **(Save)** — Saves the current file.
-  **(Cut)** — Deletes the current selection and places it on the clipboard.
-  **(Copy)** — Copies the current selection to the clipboard.
-  **(Paste)** — Pastes the contents of the clipboard to the current cursor location.
-  **(Find)** — Opens the Search dialog box, which is used to conduct an interactive search (see “Interactive Searching Using the Search Dialog Box” on page 29).
-  **(GoTo)** — Opens the **GoTo** dialog box, which is used to go to a file, line or procedure (see “Using the GoTo Dialog Box” on page 25).
-  **(Undo)** — Undoes the last edit.
-  **(Redo)** — Re-implements the last edit that was undone.
-  **(Previous File)** — Switches to the previous open file in the Editor’s stack.
-  **(Next File)** — Switches to the next open file in the Editor’s stack.
-  **(Retry Build)** — Retries building the project containing the open source file. This button is available in the Editor window that opens when a build error is encountered.
-  **(Previous Error)** — Navigates to the previous build error. This button is available in the Editor window that opens when a build error is encountered.
-  **(Next Error)** — Navigates to the next build error. This button is available in the Editor window that opens when a build error is encountered.
-  **(Save and Close)** — Saves files without prompting, then quits the Editor.

 (**Close File**) — Closes the current file.

 (**Close Editor**) — Quits the Editor. The presence or absence of this button on the toolbar is a configurable option (see the configuration option **Display close (x) buttons** in “The General Options Tab” on page 208). By default, the button does not appear.

The File and Procedure Fields

Below the toolbar are three fields you can use to navigate within and between files.

- **File** — This field displays the name and path of the file you are currently editing. You can type another filename in the field to open it in the Editor window. The drop-down menu provides access to your most recently used files. Files open in the current Editor session appear at the top, and other recently used files appear below.
- **Procedure** — The drop-down menu displays all procedures defined in the file. You can use this box to go to a procedure in the file. Select a procedure from the drop-down menu, or type a procedure name in the field. This field only appears when you are editing C or C++ files.

As you type characters in this field, the Editor will auto-complete with procedures that match the characters you type. You can type part of a procedure name, then select the drop-down menu to view a list of all procedures that match the string you have entered. If you are uncertain about a procedure’s name, you can use * and ? wildcard characters.

- **Line Number** — This box displays the line number the cursor is on. To go to a specific line in the file, type the line number in this field, and press Enter.

The Source Pane Shortcut Menu

You can right-click anywhere in the Editor source pane to open a **Shortcut Menu** of common editing operations. The shortcut menu includes some or all of the following items, depending on the cursor’s position and the type of file that is being edited.

- **Cut, Copy, Paste and Undo** — For information, see “The Edit Menu” on page 59.

- **Go To Definition** — For information, see “Go To a Definition” on page 24.
- **Go To Declaration** — For information, see “Go To a Declaration” on page 24
- **Browse References** — For information, see “Browse References” on page 24.
- **Generate Cross References** — For information, see “Generate or Regenerate Cross References” on page 25.
- **Regenerate Cross References** — For information, see “Generate or Regenerate Cross References” on page 25.
- **Show Last Edit** — For information, see “The Version Menu” on page 71.
- **Open New Editor Window** — For information, see “The File Menu” on page 56.
- **Properties** — For information, see “The File Menu” on page 56.
- **Comment, Uncomment and Auto Indent** — For information, see “The Block Menu” on page 66.

The Status Bar

The status bar is located at the very bottom of the Editor window. It displays the following information:

- **Status box** — The left corner of the status bar displays status, usage, and error messages. For example, when you press Ctrl + F (see “Incremental Searching” on page 28), the left corner of the status bar displays the search text as you type it.
- **Cursor position indicator** — Displays the current line and column position of the cursor.
- **Read-only window indicator** — A stop sign appears near the right corner of the status bar if the window permission for the current file is read-only. See also **Toggle Write Permission** in “The File Menu” on page 56 and **Read Only Window** in “The View Menu” on page 62.
- **Change dot** — A small red star appears near the right corner of the status bar if changes have been made to the file since the last time it was saved.
- **Version control status** — The letters VC appear in the right corner of the status bar if the current file is under version control. The letters will be red if the file has been modified. Otherwise they will be black.

Navigating in Files

This section provides details about Editor functions that help you navigate within a single file, and across multiple files.

Using References and Prototypes

Whenever the MULTI Editor loads a C or C++ file, it attempts to obtain information about the file, including the following information:

- The operating system for which the code was written
- The source file's include files
- The cross reference information



Note Cross reference information is available for source code in projects built using the Project Manager. Cross reference information is not available for preprocessor `#defines` in the Editor (but it is available in the Debugger).

When this information is obtained, the Editor automatically grabs the function prototypes from the include files; loads information for the operating system such as APIs, constants and types; and uses this information for syntax coloring and auto-completion (see “The language.gsc Syntax Definition Files” on page 49). The prototype information also allows you to quickly jump to the function's definition or declaration from the shortcut menu.

If cross reference information is available, it is automatically loaded (see “Generate or Regenerate Cross References” on page 25). The Editor will attempt to obtain cross references from the following sources:

- MULTI Debugger — The Editor can obtain cross reference information from the Debugger when the Editor was launched from the Debugger and the current file in the Editor is contained in the program being debugged.
- Cross reference (**.dbo**) files — These files are generated during a build.

Cross reference information for a source file may be incomplete. To obtain complete cross reference information for the source file's enclosing project, you can right-click and select **Generate Cross References** from the shortcut menu (see “Generate or Regenerate Cross References” on page 25).

Go To a Definition

To go to the definition of an item:

1. Right-click an object, function, variable, type or other text in the source pane and select **Go To Definition** from the shortcut menu.
2. The Editor will use the function prototype or cross reference information to find the definition of the clicked item and move the cursor to that position. This opens another file if the definition is not located in the current file.

Go To a Declaration

To go to a declaration of an item:

1. Right-click an object, function, variable, type or other text in the source pane and select **Go To Declaration** from the shortcut menu.
2. The Editor will use the function prototype or cross reference information to find the declaration of the clicked item and move the cursor to that position. This opens another file if the definition is not located in the current file.

Browse References

To browse reference information:

1. Right-click an object, function, variable, type, or other text in the source pane and select **Browse References** from the shortcut menu.
2. The Editor will attempt to obtain the item's cross references and display them in a **Browse** window.

The **Browse** window for cross references in the Editor is very similar to the **Browse** window in the Debugger, except that it does not have Debugger-related functions (for details, see “Browsing Cross References” in Chapter 10, “Browsing Program Elements” in the *MULTI: Debugging* book).

The status box of the **Browse** window will display the source from which the cross references are obtained:

- **Based on debugger** — the source is the symbol table of the program in the corresponding Debugger.

- **Based on progress: N/A** — the source is the single **.dbo** file of the current source file.
- **Based on progress: 70%** — the source is multiple **.dbo** files of the enclosing project. The percentage displayed represents the number of currently loaded **.dbo** files out of the total number in the enclosing project.

Cross reference **Browse** windows are accessible in the Editor through the **Windows** menu. Cross reference **Browse** windows can be deleted by selecting **View → Close Dependent Windows**.

Generate or Regenerate Cross References

To obtain complete cross reference information based on the project to which the source file belongs:

1. Right-click in the source pane and select **Generate Cross References** from the shortcut menu.



Note This menu item is unavailable if the Editor obtained cross reference information from the corresponding MULTI Debugger.

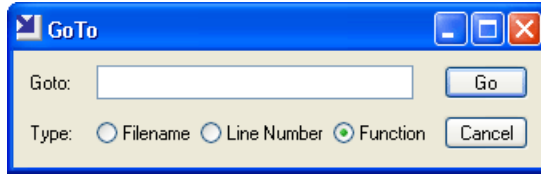
2. The Editor will search for the enclosing project and generate cross reference information for the whole project.

Generating cross reference information rebuilds the files in your program. Once cross reference information for the enclosing project is generated, the shortcut menu item changes to **Regenerate Cross References**. If any of the source files in the project have changed, the cross references for the enclosing project must be regenerated for the information to be accurate.

Using the GoTo Dialog Box

You can use the **GoTo** dialog box to go to a file, line number, or function. To open the **GoTo** dialog box, do one of the following:

- Select **Edit → Goto**.
- Click the **Goto** button ().
- Press **Ctrl + Shift + G**.



Go To a File

To open a file in the current Editor:

1. Type a filename in the **Goto** field.
2. Select the **Filename** radio button.
3. Click **Go**.

Go To a Line

To go to a specific line in the current file:

1. Type the line number in the **Goto** field.
2. Select the **Line Number** radio button.
3. Click **Go**.

Go To a Line Quickly

You can quickly go to a line without using the **GoTo** dialog box:

1. Type Ctrl+G. The **status box** in the lower left corner of the status bar will display the prompt **Goto Line:**.
2. Enter the line number to go to.
3. Press Enter.

Go To a Function

To go to a specific function in either the current file, or in another file:

1. Type a function name into the **Goto** field.

2. Select the **Function** radio button. If the **Function** button is not available, the **Editor** could not find a tag file.
3. Click **Go**.
4. If the function is in the current file, the Editor moves the cursor to the beginning of the function. If the function is in a different file, the Editor opens that file and moves the cursor to the beginning of the function.



Note This option is only available if the corresponding language is C or C++ (for which the Editor can automatically grab function prototypes from the edited file) or:

- A **ctags** style tag file named **tags** resides in the current directory. See “Using Tags in Your Files” on page 27 for more information.
- A tag file has been specified with the **Tools** → **Append TagFile** menu item.
- You use the **AppendTagFile** Editor command in the **Execute Editor Commands** dialog, described in Appendix A.

Using Tags in Your Files

If you use the **ctags** utility to create a tag file, the Editor uses the information to navigate to functions. For additional tag commands, see “Tag Commands” on page 307.

When the Editor starts, it looks for a file called **tags**. If the tag file has a different name or is in an unexpected location, the Editor may not find it. To manually load a tag file into an Editor session:

1. Select **Tools** → **Append TagFile**.
2. The **Select Tag File** dialog box appears. Select the tag file, or enter the path and filename of the tag file.
3. Click **Load**.

To reload tag files in an Editor session, select **Tools** → **Reset Tags**.

Searching in the Editor

The MULTI Editor provides three ways to search for text:

- *Incremental searching* — Performs a quick search of the active file. The search is incremental, so it searches for a string as you type it. This means that you do not always have to type the whole text string you are looking for. This type of searching uses keyboard shortcuts rather than a dialog box, so there are fewer options. For more information, see “Incremental Searching” on page 28.
- *Interactive searching* — Uses the Search dialog to perform a variety of searching tasks on the file in the active Editor window. Interactive searching allows search-and-replace and regular expression matching. For more information, see “Interactive Searching Using the Search Dialog Box” on page 29.
- *Searching files* — Performs full-text searching in all open files using the **grep** utility. For more information, see “Searching in Files” on page 33.

Incremental Searching

You can perform a quick incremental search (referred to in some command and option names as an *iSearch*) in the active Editor window without opening the Search dialog box. To do so, perform the following steps:

1. Press Ctrl + F (to search forward in the file) or Ctrl + B (to search backwards in the file). This will cause the status box in the lower-left corner of the status bar to display the prompt: **Srch**.
2. Type the string you are searching for. As you type characters, the search string is displayed in the status bar to the right of the **Srch** prompt and the first occurrence of the string pattern is highlighted in the Editor source pane. You can use the Backspace key to remove characters from the search string. As you modify the search string by typing additional characters or removing characters, the next occurrence of the new string is highlighted in the Editor source pane.

The case sensitivity of the search is set via the **Match exact case in searches** option (see “The General Options Tab” on page 208). However, note that even if the search is case-insensitive, it becomes case-sensitive if you enter any capital letters in the search string.

3. If a match has been found, press Ctrl+F again to view the next match or press Ctrl+B to view the previous match. If there is no other match, the status bar indicates that the search failed to find another match.
4. Press Esc at any time to clear the current search. (The **Srch** prompt will disappear.) The option **Escape restores view after iSearch** controls whether the cursor returns to its original location when you press Esc. For more information, see “The General Options Tab” on page 208.



Tip If you are not in search mode (that is, if the **Srch** prompt is not showing in the status bar and the Search dialog box is not up), you can press Ctrl+F two times to repeat your most recent search, whether it was an incremental search or an interactive search (that is, a search performed from the Search dialog box).

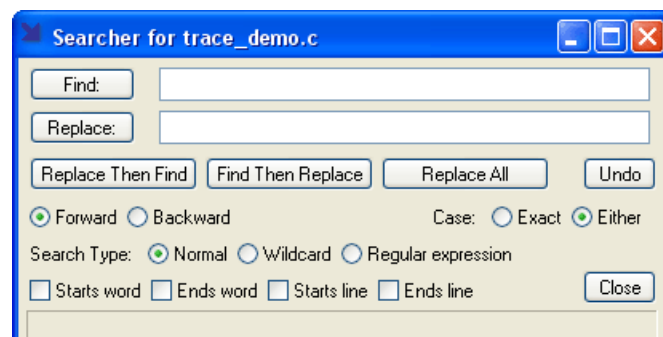
Search options that you have previously set in the Search dialog box affect incremental searches performed in the same Editor window and MULTI session. These options are: **Case**, **Search Type**, **Starts word**, **Ends word**, **Starts line**, and **Ends line**. Use the Search dialog box to clear or change these option settings. For more information about the Search dialog box and the options it contains, see “Interactive Searching Using the Search Dialog Box” on page 29.

Interactive Searching Using the Search Dialog Box

The Search dialog box allows you to specify search criteria for interactive searches.

To open the Search dialog box, do one of the following:

- Select **Edit** → **Find**.
- Click the **Find** button (.
- Press Ctrl+Shift+F.



The Search dialog box can be used to search for and to search-and-replace text in the file open in the active Editor window.

The fields and options in the Search dialog box are described in the following table.

Item	Meaning
Find	Searches for and highlights the next occurrence of the string entered in this field. To search for the next occurrence, click the Find button again, or press Enter.
Replace	Replaces the current selection with the string entered in this field.
Replace Then Find	Replaces the current selection and then searches again.
Find Then Replace	Searches for the next occurrence of the search string, and replaces it with the replacement string.
Replace All	Starts at the beginning of the file and replaces all occurrences of the search string with the replacement string.
Undo	Undoes the last Editor command. Note that this button will Undo all Editor commands, not just commands entered in the Search dialog box.
Forward Backward	Determines whether the search proceeds forward or backward from the current location. (The default is Forward.) The Editor searches from the current location in the file toward the end of the file for a forward search, and toward the beginning of the file for a backward search. If the search string is not found before it reaches the end or the beginning, the search stops and the status bar displays an error message. If you start the search again, it resumes from the beginning or the end of the file.
Case	Determines whether or not the search is case-sensitive. Select one of the following: • Exact — Selects a case-sensitive search. For example, <code>Fly</code> matches <code>Fly</code>, but not <code>fly</code> or <code>FLY</code>. • Either — Selects a case-insensitive search. For example, <code>Fly</code> matches both <code>fly</code> and <code>FLY</code>. (This is the default.)

Item	Meaning
Search Type	Determines the type of search to conduct. Select one of the following: • Normal — No special characters in the search; that is, characters only match themselves. (This is the default.) • Wildcard — The following characters have a special meaning in the search string: ? (question mark) — Matches any single character except a newline character. * (asterisk) — Matches any number of characters except newline characters. • Regular Expression — The characters listed in “Searching for Regular Expressions” on page 32 can be used in the search string.
Starts word Ends word	Specifies whether the search string must appear at the beginning or end of a word. If Starts word is checked, the search string must appear at the beginning of a word. For example, <code>fly</code> matches <code>fly</code> or <code>flybat</code>, but not <code>batfly</code>. If Ends word is checked, the search string must appear at the end of a word. For example, <code>fly</code> matches <code>fly</code> or <code>batfly</code>, but not <code>flybat</code>. If both are checked, the string must form a complete word. For example, <code>fly</code> matches <code>fly</code>, but not <code>flybat</code> or <code>batfly</code>. If neither box is checked, any occurrence of the string is found.
Starts line Ends line	Specifies whether the search string must appear at the beginning or the end of a line. These options function similarly to Starts word and Ends word, described above, except that they apply to the beginning and end of a line. Selecting both of these boxes specifies that the entire line must match the search text.
Close	Cancels the search and closes the Search dialog box.



Note Some of the settings in this dialog box set the defaults for the next incremental search (see “Incremental Searching” on page 28).

Searching for Regular Expressions

The following table lists acceptable formats for entering regular expressions as search strings when the **Regular expression** radio button has been selected in the Search dialog box.

.	(A period) Matches any single character except a new line.
[<i>string</i>]	Matches any single character appearing in <i>string</i> . For example, [abc] matches an a, b, or c. You can specify character ranges by separating the start and end of the range with a dash (-). For example, [b-e] matches b, c, d and e. To include a close bracket (]) as part of the <i>string</i> , make it either the first character of <i>string</i> , or the last character of a range. For example, []abc].
[^ <i>string</i>]	If the first character of the <i>string</i> is a caret (^), it matches any character that is not in <i>string</i> .
^	Place at the start of the search string to match the beginning of a line. Same as the Starts line option.
\$	Place at the end of the search string to match the end of a line. Same as the Ends line option.
<	Place at the start of the search string to require that the rest of the search string matches the beginning of a word. Same as the Starts word option.
>	Place at the end of the search string to require that the rest of the search string matches the end of a word. Same as the Ends word option.
(<i>regex</i>)	Matches the regular expression <i>regex</i> enclosed in parentheses.
<i>regex</i> *	Matches zero or more occurrences in succession of the regular expression <i>regex</i> .
<i>regex</i> 1 <i>regex</i> 2	Matches regular expression <i>regex</i> 1 or regular expression <i>regex</i> 2.

The following table gives some examples of regular expressions that can be used in searches.

a.d	Matches and, a d, and aud.
a.*d	Matches ad, are d, and abd.
<and	Matches and, but not stand.

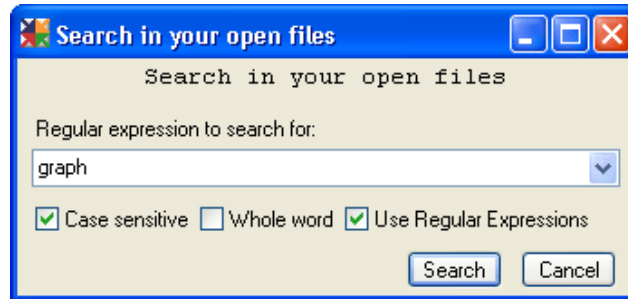
are is	Matches either are or is.
(are is)* bad	Matches are bad, is bad, areisare bad, and bad.

Searching in Files

The MULTI Editor supports full-text searching of all open files with the Search in Files feature. To search all open files, do one of the following:

- Select **Tools** → **Search in Files**.
- Enter the **grep** command in the **Execute Editor Commands** dialog, described in Appendix A.

This opens the Search in Files dialog box.



Enter the search string in the text box (or use the drop-down list to select recent search strings) and select any of the following searching options, if desired:

- **Case sensitive** — The search will only find text that matches the case of the search string exactly. If this box is not selected, the search will ignore case when searching for a match. This box is selected by default.
- **Whole word** — The search only finds text that contains the search terms as words. This means that the matching string must be preceded by a non-word character and followed by a non-word character, where word characters are letters, digits, and the underscore. For example, if you select this check box, a search for `ice` does not match `slice` or `ice__`, but it does match `ice-9`. This box is cleared by default.
- **Use Regular Expressions** — The search treats the text you enter in the text box as a regular expression. If this box is not selected, the search treats the text you enter as a fixed string. This box is selected by default.

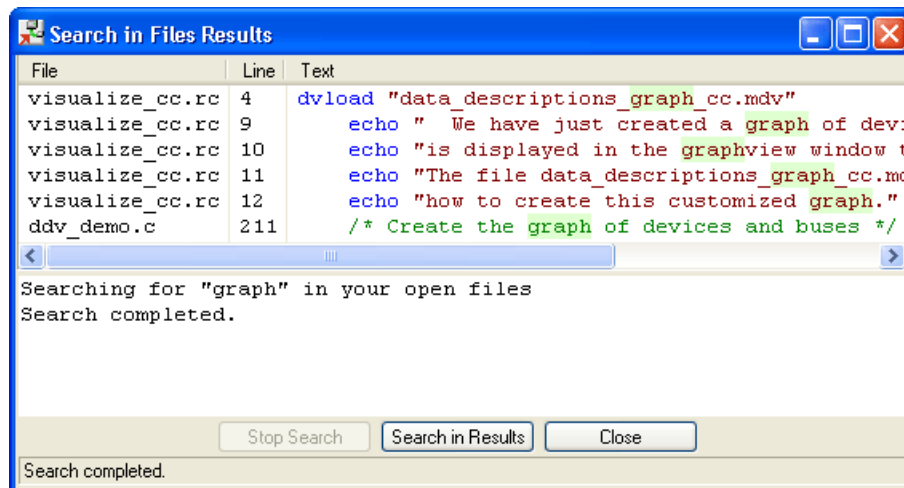
After you have entered the search string and set your searching options, click **Search** to perform the search. A **Search in Files Results** window will open and display all matches as they are found. See the next section for a description of this window.



Note The Search in Files capability works by running the BSD **grep** utility. A copy of BSD **grep** is installed along with MULTI. However, BSD **grep** is not part of MULTI and is not distributed under the same license as MULTI. For more information about the license under which BSD **grep** is distributed, refer to the file **bsdgrep.txt**, which is located in the **copyright** subdirectory of the MULTI installation directory. For information about the search expression format that BSD **grep** uses, refer to the OpenBSD `re_format(7)` man page.

Viewing Search in Files Results

The **Search in Files Results** window displays the results of a Search in Files operation.



The main components of this window include:

- Results pane — Contains columns that display information about the matching lines in the files searched.
 - **BP** column — Displays whether or not a breakpoint is set on the matching line. You can also use this column to set and remove breakpoints on certain lines. To set a breakpoint on the matching line, click the breakdot. To delete a breakpoint, click the breakpoint icon. (This column only appears if the Editor or the Search in Files dialog box has been launched from the MULTI Debugger.)

- **File** column — Displays the name of the file that contains the matching line. Place your mouse pointer over this field to display the path to the file, if available, in a tooltip.
- **Line** column — Displays the line number of the matching line.
- **Text** column — Displays the text of the matching line.
- **Text pane** — Contains a description of the search and displays any errors printed by the **grep** utility.
- **Stop Search** button — Stops the current search, but does not close the results window. When the search is complete, this button is unavailable.
- **Search in Results** button — Opens a new search window that allows you to search only in the results of your previous search. This search window is identical to the Search in Files dialog box pictured in “Searching in Files” on page 33, except that it has an additional check box labeled **Select Non-matching Lines**, which is cleared by default. If left unchecked, MULTI only searches within the individual lines displayed in the previous search window. If selected, MULTI searches within all files that contained any search results. This button appears dimmed until the initial search finishes.
- **Close** button — Closes the results window.
- **Status bar** — Displays the progress of the search. Located at the bottom of the window, this bar displays **Searching**, **Search completed**, or **Search stopped**.

You can double-click any line in the search results window to open an Editor on the specified file and line. If the Editor or the Search in Files dialog box has been launched from the MULTI Debugger, single-clicking a line displays the line in the Debugger source pane.

Working with Code

The Editor provides tools and keyboard shortcuts that you will find helpful when editing source code.

Indenting Code

Indentation is whitespace at the beginning of each line, used to denote the hierarchical structure of your code and make it more readable. As you write code, you can manually insert an indent, or you can let the Editor indent your code based on common coding standards.

Manually Inserting or Removing an Indent

To manually insert or remove an indent:

1. Move the cursor to the appropriate line, or highlight multiple lines.
2. Select **Block** → **Indent** to insert an indent.
3. Select **Block** → **Unindent** to remove an indent.

For more information about working with indents, see “The Block Menu” on page 66.

Auto Indenting Code

The MULTI Editor auto indents your code according to common coding standards.



Note Auto indenting is only available for certain languages. At present, MULTI supports auto indenting for C, C++, Java, and Ada.

1. If you want to auto indent a single line of code, move the cursor to that line. If you want to auto indent multiple lines of code, highlight those lines.
2. Select **Block** → **Auto Indent**, or press Ctrl+2 or Ctrl+; (semicolon). Pressing the Tab key also automatically indents everything to the right of the cursor.

You can make a one-time change in how far the Editor auto indents the lines of a lexical block of code by indenting the first line of code the way you want the entire block to look, then highlighting the rest of the block and starting the auto indent. This prevents the Editor from breaking the conventions of a pre-existing block.

Configuring Indents

You can configure the default size of indents and how they affect your code.

- To change the configuration of indents for all files, select **Config** → **Options** and make changes on the **MULTI Editor** tab (see “The MULTI Editor Options Tab” on page 251).
- To change the size of indents for the current file and current session only, use the **Per File Settings** dialog box. See “The Per File Settings Dialog Box” on page 64 for more information.

Auto-Indent Characters

By default, the Editor automatically makes indenting adjustments when you type the following characters:

#	number sign
*	asterisk
;	semi-colon
:	colon
{	left curly brace
}	right curly brace

You can disable characters from auto indenting both your code and comments by disabling **Auto indent as you type**. If you want special characters to auto indent your code, but want to disable them if you use them within a comment, disable **Auto indent comments as you type**. See “The MULTI Editor Options Tab” on page 251.

Working with Comments

This section describes how you can use the Editor to easily insert and manipulate comments in your code.

Inserting or Removing a Comment

The Editor uses the proper syntax for comments based on your selected programming language. Block comment operations are available to C, C++, Java, and all languages that support line comments.

- To insert a new comment, place the cursor on a blank line where you want the comment to start and select **Block → Comment**.
- To comment out existing code, select the appropriate text, then select **Block → Comment**. The selection is implicitly extended to line boundaries; the entire line the selected text is in will be commented out.
- To uncomment a comment block, highlight the block and select **Block → Uncomment**.

For more information about working with comments, see “The Block Menu” on page 66.

For information about configuring comments, see “The MULTI Editor Options Tab” on page 251.



Note If the Editor is not using the correct syntax for comments, select **View → Language** to make sure it is configured for the correct language.

Highlighting the Boundaries of the Current Block of Code

To identify the beginning and end of the current block of code, select **View → Match**.

When you perform this operation, the Editor:

1. Searches backward from the cursor and finds the first enclosing instance of a left parenthesis “(”, left curly brace “{”, or a left bracket “[”.
2. Searches forward from the cursor to find the matching ending mark.
3. Selects the code in between.

For more information, see “The View Menu” on page 62.

Working with Columns

You can cut, copy and paste columns of text in the Editor. For more information, see “The Block Menu” on page 66.

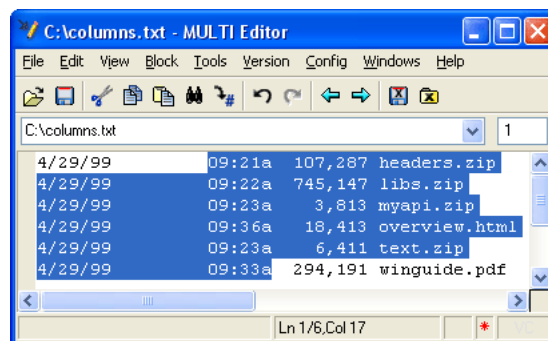
Copying a Column of Text

You can copy a column of data from a file, excluding data on either side of the column. For example, suppose you have a tab delimited file that lists date, time, file size, and filename. You can copy just the column that lists time without affecting any of the other data.

1. On the first line that contains data you want to copy, start the selection at the first character you want copy.
2. Extend the selection to include all of the data in the column. The last character selected should be the last character of the column.
3. Select **Block** → **Rect Copy**.
4. The region of text to be copied is the rectangle with the diagonal defined by the first character and the last character of the selection.

Example 1. Using Rect Copy

Highlight the following selection, then select **Block** → **Rect Copy**.



When you paste the contents of the clipboard, the following is inserted in your file:

```
09:21a
09:22a
09:23a
09:36a
09:23a
09:33a
```

Cutting a Column of Text

To cut a column of text out of a file, highlight the column and select **Block** → **Rect Cut**.

Pasting a Column of Text

After you have cut or copied a column of text from a file, you can:

- Select **Block** → **Rect Paste** to paste the column without inserting line breaks.
- Select **Edit** → **Paste** to paste the contents of the clipboard into the file with line breaks.

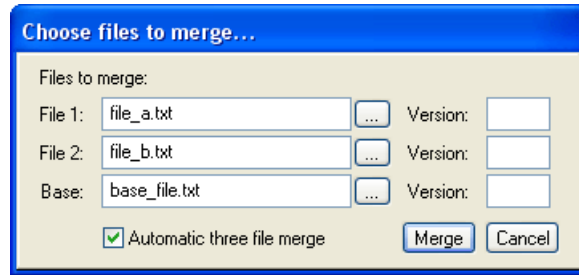
Working with Multiple Files

This section provides information about the MULTI Editor tools you can use to merge and compare multiple files.

Merging Files

You can use the Editor to merge two or three files into a single file. The initial files can be completely different files, or different versions of the same file.

To begin merging files, select **Tools** → **Merge Files** to open the **Choose files to merge** dialog box.



The **Choose files to merge** dialog box contains the following fields and buttons:

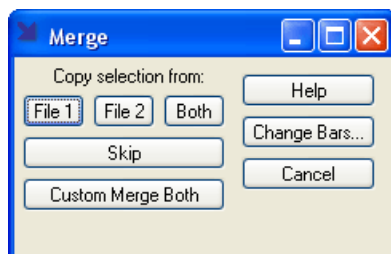
File 1	Type the name of the first file you want to merge, or click  (Browse) to select a file. If you are using a version control system and want to merge a different version of a file from disk, enter the filename in the File 1 field and the version of that file to be merged in the adjacent Version field.
File 2	Type the name of the second file you want to merge, or click  to select a file. If you specify the same file in the File 1 and File 2 fields without specifying a version, then File 1 refers to the copy currently open in the Editor, while File 2 refers to the file on disk.
Base	Type the name of the file from which the other two files, File 1 and File 2, are derived, or click  to select a file. If you are only merging two files, leave this field blank.
Automatic three file merge	Select this box if you want the Editor to attempt to resolve three-file merges without prompting you. Note that even with this option selected, the Editor prompts you if it encounters a conflict that it cannot resolve. Clear this box if you want to manually review every proposed merge.
Merge	Click Merge to begin the merge. The two-file Merge dialog box will open, or, if you specified a file in the Base field, the three-file Merge dialog box will open. See “Merging Two Files” on page 42 or “Merging Three Files” on page 43.

Merging Two Files

To merge two files:

1. Select merge criteria in the **Choose files to merge** dialog box (see “Merging Files” on page 40) and click **Merge**. An Editor window appears for each of the two files you specified, as well as an extra **Result** window that displays results of the merge. The **Status box** in the lower left corner of each window identifies which file is in each window.

In addition to the Editor windows, the two-file **Merge** dialog box will open:



The Editor will pause at each difference between the files and highlight the text that is different.

2. Use the following buttons in the two-file **Merge** dialog box to control the merge results:

File 1	Copies the selected text from File 1 into the Result window.
File 2	Copies the selected text from File 2 into the Result window.
Both	The first time you click this button, the dialog box Configure change bars for this merge session appears (see “Configure Change Bars for This Merge Session” on page 45). The next time you click this button, the last values entered are used.
Skip	Finds the next difference without adding the current difference to the file.
Custom Merge Both	Performs the same action as the Both button, except that you can modify the changes in a temporary window before merging them into the new file.
Help	Opens MULTI’s online help for the Merge dialog box.
Change Bars	Changes the settings for the Both button. When you click this button, it opens the same dialog box that appears the first time you click the Both button (see “Configure Change Bars for This Merge Session” on page 45).
Cancel	Aborts the merge, closing all merge windows.

The results of each merge selection will be copied to the **Result** window. You can also manually cut and paste text into the **Result** window.

3. When the merge is complete, a **Save** dialog box opens so you can save the **Result** file.

After you save the results, the other Editor windows close. If you save the **Result** file with the same name as one of the original files, any Editor windows still open on that file will be updated with the merged results.

Merging Three Files

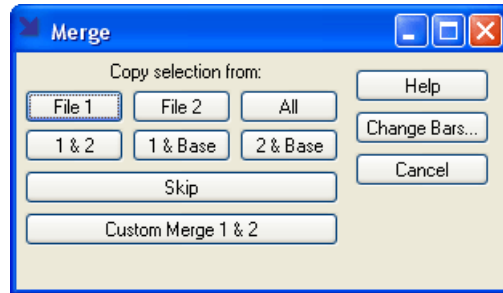
When you merge three files, one file is considered the **Base** file from which the other two are derived. With this assumption, the Editor is usually able to merge without asking you for information, using the following rules:

- If a difference exists between the two source files, and one is the same as the **Base** file, then the Editor uses the one that is different from the **Base** file.
- If both source files differ from the **Base** file, but are the same as each other, then the Editor uses the new text from either source file.
- If all three files are different, a conflicting change was made and the Editor has to ask which change to use. In this case, it is likely that you will have to merge the change manually.

To merge three files:

1. Select merge criteria in the **Choose files to merge** dialog box (see “Merging Files” on page 40) and click **Merge**. An Editor window appears for each of the three files you specified, as well as an extra **Result** window that displays results of the merge. The **Status box** in the lower left corner of each window identifies which file is in each window.

In addition to the Editor windows, the three-file **Merge** dialog box will open.



The Editor will pause at each difference between the files and highlight the text that is different.

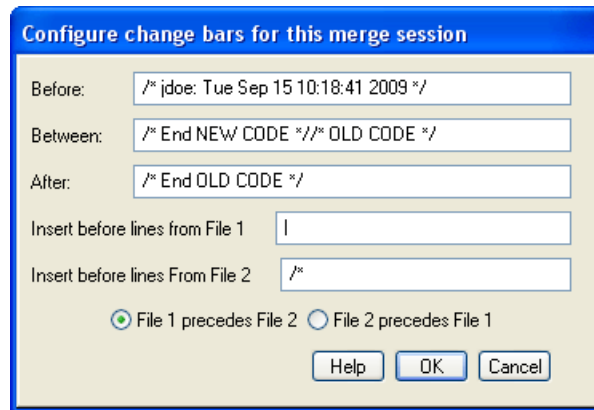
2. Use the following buttons in the three-file **Merge** dialog box to control the merge results:

Button	Meaning
File 1	Copies the selected text from File 1 into the Result window.
File 2	Copies the selected text from File 2 into the Result window.
All	Copies the selected text from all three files. The first time you click this, the dialog box Configure change bars for this merge session appears (see “Configure Change Bars for This Merge Session” on page 45). The next time you click this button, the last values entered will be used.
1 & 2	Merges the selections from File 1 and File 2 . The first time you click this, the dialog box Configure change bars for this merge session appears (see “Configure Change Bars for This Merge Session” on page 45). The next time you click this button, the last values entered will be used.
1 & Base	Functions the same as the 1 & 2 button, except it merges the selections from File 1 and the Base file.
2 & Base	Functions the same as the 1 & 2 button, except it merges the selections from File 2 and the Base file.
Skip	Finds the next difference without adding the current difference to the results file.
Custom Merge 1 & 2	Performs the same action as the 1 & 2 button, except that you can modify the changes in a temporary file before merging them into the new file.
Help	Opens MULTI’s online help for the Merge dialog box.

Button	Meaning
Change Bars	Changes the settings for the 1 & 2 , 1 & Base , 2 & Base , and All buttons. This button opens the same dialog box that appears the first time you click any of those buttons (see “Configure Change Bars for This Merge Session” on page 45).
Cancel	Aborts the merge, closing all merge windows.

Configure Change Bars for This Merge Session

The dialog box **Configure change bars for this merge session** asks you how you want to merge multiple selections (with comments in front, between, or after them; with change bars around them; in what order; and so forth).

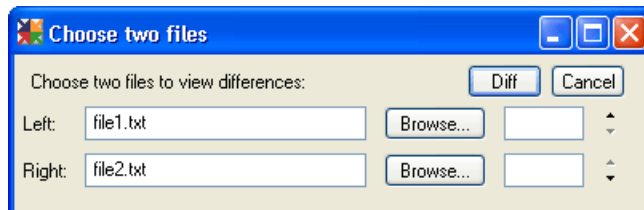


This dialog box is accessible from the **Merge** dialog box (see “Merging Two Files” on page 42 or “Merging Three Files” on page 43).

Comparing Files

You can compare files without merging them. To compare two files, do the following:

1. Select **Tools** → **Diff Files**. The **Diff Files** dialog box will open:



2. In the **Left** field, specify the *filename* and *version* of the first file you want to compare. Click the up or down arrows to increment or decrement the version number. If you do not specify a version, the Editor assumes you mean the current version.
3. In the **Right** field, specify the *filename* and *version* you want to compare to the first file. Click the up or down arrows to increment or decrement the version number. If you do not specify a version, the Editor assumes you mean the current version.
4. Click **Diff**.

The Editor opens a **Diff Viewer** that shows the differences between the two specified versions. For more information, see “The Diff Viewer” on page 157.

Configuring the Editor

You can change how the Editor looks and functions to help you work more efficiently. Many of the options that control how the Editor appears and handles your code are accessible by selecting **Config** → **Options**, then the **MULTI Editor** tab. For more information about configuring MULTI, see Chapter 8, “Configuring and Customizing MULTI” and Chapter 9, “Configuration Options”.

Customizing the Editor GUI

You can customize the Editor to perform actions in new ways. You can replicate any of the actions available in the standard Editor window by assigning commands to new menu items, buttons, key bindings, or mouse click combinations. For example, if you prefer not to use the mouse when you are editing, you can assign commands to keystrokes; or if there is a menu item that you access frequently, you can assign the equivalent commands to a button.

- For information about how you assign commands to menus, key bindings, mouse bindings, and buttons, see “Customizing the GUI” on page 181 in Chapter 8, “Configuring and Customizing MULTI”.
- For information about Editor commands, see Chapter 3, “Editor GUI Reference”, which lists the equivalent commands for GUI menu items; and Appendix A, which provides a complete list of Editor commands.
- For a complete list of the default key and mouse bindings, see Appendix C.

Configuring the Editor for a Programming Language

The Editor automatically configures itself based on the programming language of the source file that is currently being edited. For example, if you open a file **foo.c**, the Editor automatically identifies text enclosed in double quotes (" ") as a string, completes certain words (such as language-specific keywords) as they are typed into the file, and automatically colors the code to make it more readable.

Two files are used to identify the programming language in a source file, color the programming language, and enable auto-completion:

- **index.gsc** — The index configuration file that the Editor uses to match the source file with the appropriate syntax definition file.
- **language.gsc** — The syntax definition file for a specific programming language. This file defines how the Editor should color the language and whether the Editor should auto-complete certain words as they are typed.

Adding Support for New Languages

You can create new syntax definition files for any language, for example you can add support for a proprietary scripting language. To implement support, you must create a syntax definition file for the language (for example, *my_language.gsc*), and then create an entry in the **index.gsc** file for the new language.



Tip MULTI's default index configuration file and syntax definition files include detailed comments. The easiest way to add a new language is to look at the default files, which are located in one of the following directories:

- Windows — *install_dir\defaults\syntax_coloring*
- UNIX — *install_dir/defaults/syntax_coloring*

The index.gsc Configuration File

When the Editor opens a file, it searches the **index.gsc** configuration file for the file's extension and uses it to locate the corresponding language syntax definition file. The **index.gsc** configuration file is located in the **defaults/syntax_coloring** directory of your MULTI installation.

Each entry defines the basic information about a language. The definition order determines how they will be displayed in the Editor's **View → Language** menu or submenu. For each language, the **index.gsc** file defines the following:

Identifier
A unique identifier for the name of the language. This identifier must correspond to the <code>name</code> in the language's syntax definition file (see "The language.gsc Syntax Definition Files" on page 49). If the language's identifier contains any non-alphanumeric characters, enclose the entire identifier in quotes, for example: "C++".
extension
Lists the file extensions of files that contain the language. To force the Editor to automatically use a language that is not consistent with a file's extension, insert <code>-*- language -*-</code> on the first line of the file (include this syntax within a comment). For example, to force the Editor to view a file as C++, regardless of its extension, enter: <code>// -*- c++ -*-</code> on the first line of the file.
description
The name of the language as it will appear in the Editor's View → Language menu. The <code>&</code> symbol indicates that the following key is a hot key for the menu item. For example, if you specify <code>description = "Per&l"</code>, the letter <code>l</code> becomes the hot key.
definition_file
The language's syntax definition file. If an absolute path is not specified, the Editor will look for the syntax definition file in the syntax_coloring directory of your local or site-wide MULTI configuration directory. For more information, see "The language.gsc Syntax Definition Files" on page 49.
hide
You can determine whether the language appears in the Editor's View → Language menu. To hide a language, enter <code>hide = true</code>. The default is <code>false</code>. Although a hidden language will not appear in the menu, the Editor will still use it to color files written in the corresponding language.


```
submenu
```

To group languages into a submenu within the **View** → **Language** menu, enter:

```
submenu = submenu_name
```

```
first_line_pattern
```

The regular expressions that identify the language by the source file's first non-blank line. If the Editor cannot match a language based on the source file's extension, or if the extension matches more than one entry in **index.gsc**, the Editor will use these regular expressions to select the correct language. Be sure to use a double escape character sequence (\\) when defining the regular expressions.

Regular expression support is provided by the PCRE library package, which is open source software, written by Philip Hazel, and copyrighted by the University of Cambridge, England. It can be downloaded from the University of Cambridge FTP server (<ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/>).

Example 2. The index.gsc Entry

The entry in the index configuration file for the Perl language might look like:

```
Perl {
    extension = {"pl"}
    description = "Perl"
    definition_file = "perl.gsc"
    first_line_pattern = {"^[\\s]*#![\\w /\\\\]*[\\w\\\\]perl"}
}
```

To visually separate languages when they are listed in the **View** → **Language** menu, define a unique identifier with a description of "\\x01". An example follows.

```
"GHS-Sep1" {
    description = "\\x01"
}
```

Each separator should be given a unique name.

The language.gsc Syntax Definition Files

The MULTI Editor uses *language.gsc* syntax definition files to determine items such as the display color of language elements, case sensitivity, and auto-completion. Each language has its own syntax definition file located in

the **syntax_coloring** directory. The definitions for some languages are split into multiple files, in which case the *language.gsc* file can use `%include "filename.gsc"` statements to include definitions from other syntax definition files.

Each of the following elements can be defined in the syntax definition file. For each element that can be defined, you can use the default color or specify your own.

general

The following entries define general settings for the language:

- `name` = The identifier for the language. This identifier must match the Identifier listed in the **index.gsc** file (see “The index.gsc Configuration File” on page 47).
- `description` = The name of the language as it will appear in the Editor's **View** → **Language** menu.
- `extension` = File extensions associated with this language.
- `case_sensitive` = Defines whether the language is case-sensitive. For a case-insensitive language, enter `case_sensitive = false`. The default is `true`.
- `pascal_style_priority` = (Boolean) In languages such as C/C++, comments, strings and characters receive higher priority than preprocessor statements; while other languages, such as Pascal, give higher priority to preprocessor statements. To assign priority to preprocessor statements, enter `pascal_style_priority = true`. The default is `false`.
- `separator` = Separators used to delimit syntax tokens.
- `escape` = Escape sequence leader.

For example:

```
name = "Pascal"
description = "Pascal"
extension = {"p", "pas", "h"}
case_sensitive = false
separator = "\\\"+-*/<=>:,;'\t() []^%!~|& {} .@?"
escape = "\\\"
```

preprocessor

Specifies preprocessor keywords. Limitations are:

- There can only be one non-alphabetic preprocessor lead character. Example lead characters include `%` and `#`.
- Semantics are C-like.

keyword	Defines a list of keywords, the display color for keywords, and/or whether keywords are auto-completed.
line_comment	Specifies line comment characters. Multiple line comment leaders can be defined, so an index number is used for each. For example: 1 = "//"
comment	Specifies comment characters. Multiple comment characters can be defined, so an index number is used for each. For example: 1 { begin = "(*"; end = "*)" } 2 { begin = "{"; end = "}" }
string	Specifies string characters. Multiple string characters can be defined, so an index number is used for each. For example: 1 { begin = "\""; end = "\"" }
character	Specifies constant characters. Multiple characters can be defined, so an index number is used for each. For example: 1 { begin = "\"'"; end = "\"'" }
block	Specifies a block. A block must contain a <code>begin</code> and an <code>end</code> string. You can use special characters to identify the column position in a line. The <code>begin</code> string <code>\x01</code> specifies that you begin at the beginning of line 1. The <code>end</code> string <code>\x02</code> specifies that you end at the beginning of line 2.
integer	The following entries can be made to define integers: • <code>hex</code> = Specifies support for hex integers. To support hex integers, enter <code>hex = true</code>. The default is <code>false</code>. • <code>case_sensitive</code> = Specifies case sensitivity in integer suffixes. To make integer suffixes case-insensitive, enter: <code>case_sensitive = false</code>. The default is <code>true</code>. • <code>decimal_suffix</code> = Specifies a list of suffixes for decimal integers. • <code>hex_suffix</code> = Specifies a list of suffixes for hex integers.

`float`

The following entries can be made to define the display of numbers in floating point form.

- `scientific` = Specifies whether scientific format is supported. To use scientific format, enter `scientific = true`. The default is `false`.
- `case_sensitive` = Specifies case sensitivity for the float suffix. The default is `true`. To override the default, enter `case_sensitive = false`
- `suffix` = Specifies a list of suffixes for the float.

`customized`

Defines a list of customized patterns (wildcards can be used in them), and whether any customized patterns not containing wildcards are auto-completed.

`autocomplete`

Defines the auto-complete behavior. For information, see “Configuring Auto-Complete” on page 52.

Configuring Auto-Complete

A language syntax definition file determines whether keywords, preprocessor statements, and customized items will be automatically completed by the Editor as they are typed. Most of the language syntax definition files included with MULTI have auto-complete enabled for both keywords and preprocessor statements. Auto-complete can also be enabled for function prototypes.



Note Auto-completion does not work for customized items that contain wildcard characters.

You can turn auto-complete on or off for an entire language, or for a specific category of items in the syntax definition file by specifying `autocomplete = false` or `autocomplete = true`.

The following sections provide details about auto-complete. You can modify auto-complete by editing the `autocomplete` section of the syntax definition file.



Note Even when auto-completion is disabled, you can use the keyboard shortcuts listed in “Auto-Completion” on page 333 to perform auto-completion functions based on the characters located at the cursor.

First Match vs. Best Match Auto-Complete

Auto-completion uses either *first match* logic or *best match* logic to complete words based on the letters that have been typed:

- `autocomplete = "first"` — Completes the first word that begins with the letter that has been typed.
- `autocomplete = "best"` — Does not complete a word until it can uniquely identify the letters that have been typed. This is the default setting.

For example, assume auto-completion has been enabled for the keywords: `main`, `mailman` and `mist`. When the letter `m` is typed, first match logic auto-completes the item into `main`, but best match logic does not perform any auto-completion. When the letters `ma` have been typed, first match logic still auto-completes the item into `main`, while best match logic auto-completes the item to `mai` because it still cannot determine whether the item should be `main` or `mailman`.

Minimum String Length

You can control the minimum number of characters in a string before auto-complete attempts to match the string.

The default value is 1. To change this setting, enter:

```
min_string_length = num
```

where *num* is the number of characters in the string.

Auto-Completion of Function Prototypes

Auto-complete entries also control the use of function prototypes. Function prototypes are often defined in included files. Whenever the Editor loads a C or C++ file, it automatically grabs the include files for function prototypes if it can find the include files.

- `grab_prototype` = Specifies whether function prototypes are dynamically grabbed as you type in the Editor. The default setting is `false`. To dynamically grab function prototypes as you type, enter `grab_prototype = true`. Automatically grabbing function prototypes is only supported for C and C++.

- `show_prototype` = Specifies whether function prototypes are displayed as you type. To automatically display function prototypes, enter `show_prototype = true`. When when you type a function name in the Editor followed by a “(” (open parentheses), a tooltip containing prototype information will appear to guide you through the function’s arguments. The default setting is `false`.

These two options are explicitly enabled in the syntax definition files for C and C++.

Editor GUI Reference

Chapter 3

This Chapter Contains:

- The File Menu
- The Edit Menu
- The View Menu
- The Block Menu
- The Tools Menu
- The Version Menu
- The Config Menu
- The Windows Menu
- The Help Menu
- UNIX Dialog Boxes

The MULTI Editor contains menus, toolbar buttons, and keyboard shortcuts for editing files.

All of the Editor GUI menus are described in the tables that follow. The tables also include the equivalent toolbar buttons and keyboard shortcuts, where applicable. For a complete list of button descriptions, see “The Editor Toolbar” on page 20. For a complete list of keyboard shortcuts, see “Default Keyboard Shortcuts” on page 324.

Most of the menu items also have equivalent commands, which may be entered by using the GUI menu item **Tools** → **Execute Editor Commands** (see “The Tools Menu” on page 68). Advanced users can also use these commands to configure their keyboard or mouse shortcuts (see “Customizing Keys and Mouse Behavior” on page 191). For a complete list of Editor commands, see Appendix A.

The File Menu

The **File** menu contains the following items:

New Editor (Ctrl + N)

Opens a file in a new Editor window.

Select a file from the Edit File dialog box and click Edit ; or to create a new file, enter the new filename and click Edit .
--

This menu item is bound to the LoadFileWithNewEditor command (see “File Commands” on page 285).
--

Open (Ctrl + O)
--

Opens a file in the current Editor window.
--

Select a file from the Edit File dialog box and click Edit ; or to create a new file, enter the new filename and click Edit . The selected file is pushed to the top of the context stack for the current window.
--

This menu item is bound to the OpenFile command (see “File Commands” on page 285).

Save (Ctrl + S)
--

Saves the current file.

This menu item is bound to the Save command (see “File Commands” on page 285).

Save As (Ctrl + Shift + S)

Saves the current file with a new name. Specify the new path and filename in the **Save As** dialog box.

This menu item is bound to the **SaveAs** command (see “File Commands” on page 285).

Save All

Opens the **Save All** dialog box. All open files with changes that have not been saved will be listed. Select the check boxes next to the files you want to save, then click **Save Selected**. To save all of the listed files, click **Save All**.

This menu item is bound to the **QuerySaveAll** command (see “File Commands” on page 285).

Toggle Write Permission

Toggles the permission for the file between read-only and read/write mode, if possible. A stop sign in the right corner of the status bar indicates that the file is currently read-only. This menu item changes both the file’s window permission and the file’s permission on disk. See also the **Read Only Window** menu item in “The View Menu” on page 62.

This menu item is bound to the **ToggleWritePermission** command (see “View Commands” on page 315).

Revert to Saved

Undoes all changes made to the current file since the last time it was saved.

This menu item is bound to the **Revert** command (see “File Commands” on page 285).

Revert to Backup

Undoes all changes made to the current file since the last time it was backed up. For information about enabling backups, see the configuration option **Create backup files when saving** in “The MULTI Editor Options Tab” on page 251.

This menu item is bound to the **RevertToBackup** command (see “File Commands” on page 285).

Page Setup

Windows only

Opens the **Page Setup** dialog box, which allows you to set printing options for the current file.

This menu item is bound to the **PageSetup** command (see “File Commands” on page 285).

Print

Opens the **Print** dialog box, which allows you to print the current file to a printer or to a file. For information about the UNIX **Print** dialog box, see “The Print Setup Dialog Box” on page 77.

This menu item is bound to the **Print** command (see “File Commands” on page 285).

Properties

Displays information the Editor is able to obtain about the current file, including:

- Full file path
- Language
- Number of include files
- Cross reference file
- Extra syntax information
- Enclosing project name
- Progress information about the cross reference files for the enclosing project

This menu item is bound to the **FileProperties** command (see “File Commands” on page 285).

1, 2, 3, 4, 5, 6, 7, 8

Lists a maximum of eight previously viewed files. Select any of the files to open it in the current Editor window.

Close File (Ctrl + Shift + P)

Closes the current file and prompts you to save any unsaved changes. If the file was checked out of version control during this session, you will be prompted to check it back in.

This menu item is bound to the **Close** command (see “File Commands” on page 285).

Close Editor (Ctrl + Q)

Prompts you to save the changes made to open files, then exits the Editor.

This menu item is bound to the **Close** command (see “File Commands” on page 285).

Exit All

Prompts you to save the changes made to all open files, then quits all MULTI IDE tools.

This menu item is bound to the **Quit** command (see “File Commands” on page 285).

The Edit Menu

The **Edit** menu contains the following items:

Undo (Ctrl + Z)

Reverts the last change made to the current file. Each **Undo** reverts one more change. This process can be repeated until all changes made since the file was opened are undone.

This menu item is bound to the **Undo** command (see “Undo/Redo Commands” on page 312).

Redo (Ctrl + Y)

Restores the last edit that was removed by **Undo**. You can **Redo** multiple **Undos**, until you make new changes to the file.

This menu item is bound to the **Redo** command (see “Undo/Redo Commands” on page 312).

Repeat Last Edit (Ctrl + .)

Repeats the last edit you made to the file.

You can only use this feature with certain types of edits: text input or replacement. For example, if you just selected text and replaced it with new text, then **Repeat Last Edit** will delete a similar selection contiguous to the cursor and insert the new text.

For example, your file contains the text:

```
This is a block of text.
```

You select the word `block` and type `string`, to change the text to:

```
This is a string of text.
```

If you move the cursor to the beginning of the word `text` (you do not have to highlight the word) and select **Edit** → **Repeat Last Edit**, the text will change to:

```
This is a string of string.
```

This menu item is bound to the **RepeatLast** command (see “Undo/Redo Commands” on page 312).

Cut (Ctrl + X)

Deletes the current selection and places it on the clipboard.

This menu item is bound to the **Cut** commands (see “Clipboard Commands” on page 279).

Copy (Ctrl + C)

Copies the current selection to the clipboard.

This menu item is bound to the **Copy** commands (see “Clipboard Commands” on page 279).

Paste (Ctrl + V)

Pastes the clipboard contents to the current location.

This menu item is bound to the **Paste** commands (see “Clipboard Commands” on page 279).

Delete (Ctrl + D)

Deletes the current selection. If there is no current selection, deletes the character after the insertion point.

This menu item is bound to the **Delete** command (see “Text Deletion Commands” on page 309).

Select All (Ctrl + A)

Selects the entire contents of the current file and moves the cursor to the end of the file.

This menu item is bound to the **SelectAll** command (see “Selection Commands” on page 302).

Find (Ctrl + Shift + F)

Opens the Search dialog box which allows you to set criteria for an interactive search. For more information, see “Interactive Searching Using the Search Dialog Box” on page 29.

Alternately, use Ctrl + F to perform an incremental search for a string without using the dialog box. For more information, see “Incremental Searching” on page 28.

These items are bound to the **Search** and **iSearch** commands (see “Search Commands” on page 301).

Goto (Ctrl + Shift + G)

Opens the **GoTo** dialog box, which allows you to go to a file, line number or function. For more information, see “Using the GoTo Dialog Box” on page 25.

Use Ctrl + G to quickly go to a line number without using the dialog box. For more information, see “Go To a Line Quickly” on page 26.

These items are bound to the **Goto** and **EditLine** commands (see “Navigation Commands” on page 296).

DOS Format

Toggles the file format between DOS format and UNIX format. A check mark next to this menu item indicates that the current file format is DOS. (The DOS format saves the file with carriage returns.)

This menu item is bound to the **DosFormat** command (see “File Commands” on page 285).

The View Menu

The **View** menu contains the following items:

Language

Specifies the programming language used in the current source file. The Editor uses this setting for syntax coloring, commenting, and auto-indenting features.

This menu item is bound to the **SelectLanguage** command (see “File Commands” on page 285).

Per Language Settings

Opens the **Language Settings** dialog box which lists settings for the selected language. The items in this dialog box are:

- **Auto-Complete** — Indicates whether auto-completion is active for the selected language. For more information, see “Configuring Auto-Complete” on page 52.
- **Auto-completion Match Algorithm** — Select **Best Match** or **First Match**
- **Min String Length for Auto-completion** — The number of characters to be entered before auto-completion attempts to match the string.
- **Max Number of Objects to Show for Auto-completion** — The number of matched objects to display when you enter Ctrl + / or Ctrl + ' to show the matched object names or function prototypes.
- **Dynamically Grab Function Prototype** — For C and C++ code, the Editor can automatically grab the function prototypes in the edited files. The default is specified in the corresponding language’s configuration file (see “The language.gsc Syntax Definition Files” on page 49).
- **Show Function Prototype** — If checked, the Editor displays a function’s prototype when you type in a function name followed by (. The default is specified in the language’s configuration file (see “The language.gsc Syntax Definition Files” on page 49).

The setting defaults are specified in the corresponding language’s configuration file (see “The language.gsc Syntax Definition Files” on page 49).

This menu item is bound to the **LanguageOptions** command (see “File Commands” on page 285).

Per File Settings

Opens the **Per File Settings** dialog box which lists variables that can be set for an individual file. These settings are used only for the current session. See “The Per File Settings Dialog Box” on page 64 for details.

This menu item is bound to the **EditorFlags** command (see “File Commands” on page 285).

Next File (Ctrl + Shift + Tab)

Switches to the next open file in the Editor stack.

This menu item is bound to the **CyclePushBack** command (see “View Commands” on page 315).

Previous File (Ctrl + Tab)

Switches to the previous open file in the Editor stack.

This menu item is bound to the **CyclePush** command (see “View Commands” on page 315).

Flash Cursor (Esc)

Scrolls to and flashes the line containing the cursor.

This menu item is bound to the **FlashCursor** command (see “Navigation Commands” on page 296).

Match (Shift + Right-click)

Searches backward from the cursor for the first *paired character* (parenthesis, square bracket, or curly brace) at the same nesting level as the cursor, and selects the paired characters and all text enclosed by them.

This menu item is bound to the **SelectToMatch** command (see “Selection Commands” on page 302).

Column

Opens the **Column** dialog box, which allows you to move the cursor to the specified column on the current line, if that column exists.

This menu item is bound to the **Column** command (see “Navigation Commands” on page 296).

Read Only Window

Toggles the window permission for the file between read-only and read/write mode, if possible. A check next to this menu item indicates that the Editor treats the file as read-only. This does not affect the file’s permission on disk. See also the **Toggle Write Permission** menu item in “The File Menu” on page 56.

This menu item is bound to the **ToggleReadOnly** command (see “File Commands” on page 285).

Go To Definition of Selected Object

Displays the source code, if available, for the object’s definition in the source pane. This menu item only appears when you are editing C, C++, or Ada files.

This menu item is bound to the **GotoObjDef** command (see “View Commands” on page 315).

Go To Declaration of Selected Object

Displays the source code, if available, for the object's declaration in the source pane. This menu item only appears when you are editing C, C++, or Ada files.

This menu item is bound to the **GotoObjDecl** command (see "View Commands" on page 315).

Browse References of Selected Object

Displays the object's cross references, if any, in a Browse window. For more information, see "Browse References" on page 24. This menu item only appears when you are editing C, C++, or Ada files.

This menu item is bound to the **BrowseObjXRef** command (see "View Commands" on page 315).

Generate Cross References / Regenerate Cross References

Obtains complete cross reference information based on the project to which the source file belongs. When you select this menu item, the Editor will search for the enclosing project and generate cross reference information for the whole project.

Once cross reference information for the enclosing project is generated, the menu item will change to **Regenerate Cross References**. If any of the source files in the project have changed, the cross references for the enclosing project must be regenerated for the information to be accurate.

These menu items only appear when you are editing C or C++ files.

These menu items are bound to the **GenerateXrefInfo** and **RegenerateXrefInfo** commands.

Close Dependent Windows

Deletes all cross reference **Browse** windows.

This menu item is bound to the **CloseDependentWindows** command (see "View Commands" on page 315).

The Per File Settings Dialog Box

This dialog box lists a number of indenting and text wrapping options that you can set for the current file and current session only.

To open this dialog box, select **View → Per File Settings**.



Note These settings default to the Editor options in **Config → Options**. For more information about these settings, see "The MULTI Editor Options Tab" on page 251 and "The More Editor Options Dialog Box" on page 257.

The **Per File Settings** dialog box contains the following fields:

Indent size
Controls the number of spaces the Editor inserts when you press the Tab key. The default is 4 spaces. If you enter 0 to disable tabs, pressing Tab will insert a single space.
Ada indent size
For Ada language file types only. Sets the indentation size. The default is 3 spaces. For more information, see “The MULTI Editor Options Tab” on page 251.
Ada continuation size
For Ada language file types only. Determines how far a line of Ada source code is indented if it continues on multiple lines.
Wrap column
If Word wrap is enabled, this setting specifies the column at which word wrap takes effect. If a line grows to longer than this number of columns as you are typing, the Editor will insert a line break at the first whitespace character before the column. It will indent the newly formed line by the same amount as the line above it, plus the wrap indent offset, described below.
Wrap indent offset
If Word wrap is enabled, this setting determines how much the wrapped line will be indented past the indentation of the line above it.
Word wrap
Check this box to enable word wrap, or clear it to disable word wrap. If Word wrap is enabled, the Editor will wrap lines automatically as you type, but will not wrap long lines while loading a file.
Disk format
Determines whether the file will be saved in UNIX format or DOS format (with carriage returns). It defaults to the appropriate setting.
Procedure list order
Selects the order in which procedures will display in the Procedure drop-down menu (see “The File and Procedure Fields” on page 21). Permitted settings for this option are:
• Base Name — Displays procedures sorted by the procedure’s base name. • Full Name — Displays procedures sorted by their class name, then procedure name. • Position — Displays procedures in the order they appear in the file.

The Block Menu

The **Block** menu contains the following items:

Indent (Ctrl + i)

Indents at the beginning of the current line or selected lines. The default size is four spaces. To change the indent size, select **Config** → **Options**, then select the **MULTI Editor** tab and edit the **Indent size** field.

This menu item is bound to the **Indent** command (see “Indentation Commands” on page 292).

Unindent (Ctrl + Shift + i)

Deletes a number of spaces equal to or less than the size of an indent from the beginning of the current line.

This menu item is bound to the **Unindent** command (see “Indentation Commands” on page 292).

Auto Indent (Ctrl + 2)

Indents the current line or block of lines to the position indicated by the syntax of the code. This option is only available for the C, C++, Java, and Ada languages. For more information, see “Auto Indenting Code” on page 36.

This menu item is bound to the **AutoIndent** command (see “Indentation Commands” on page 292).

Comment (Ctrl + *)

Inserts language specific characters to signify that the selected text is a comment, and not code.

This menu item is bound to the **CommentBlock** command (see “Block Commands” on page 277).

UnComment (Ctrl + Shift + U)

Removes comment characters from the selected text to make it active code.

This menu item is bound to the **UnCommentBlock** command (see “Block Commands” on page 277).

UpperCase (Ctrl + +)

Changes all the characters in the current selection to uppercase.

This operation is not supported for Green Hills Script.

This menu item is bound to the **UpperCaseBlock** command (see “Block Commands” on page 277).

LowerCase (Ctrl + -)

Changes all the characters in the current selection to lowercase.

This operation is not supported for Green Hills Script.

This menu item is bound to the **LowerCaseBlock** command (see “Block Commands” on page 277).

Rect Copy

Copies a rectangular subsection of the current selection to the clipboard. For more information, see “Working with Columns” on page 39.

This menu item is bound to the **RectCopy1** command (see “Clipboard Commands” on page 279).

Rect Cut

Deletes a rectangular subsection of the current selection, and copies it to the clipboard. For more information, see “Working with Columns” on page 39.

This menu item is bound to the **RectCut1** command (see “Clipboard Commands” on page 279).

Rect Paste

Pastes a clipboard selection created with **Rect Copy** or **Rect Cut** to the current location without line breaks. If you use **Edit** → **Paste** (Ctrl + V) to paste a rectangular selection, line breaks will be added. For more information, see “Working with Columns” on page 39.

This menu item is bound to the **RectPaste1** command (see “Clipboard Commands” on page 279).

Cut Lines (Ctrl + M)

Extends the current selection to the closest line boundaries, deletes the selection, and places it on clipboard number one.

This menu item is bound to the **SelectLine; Cut1** command combination (see “Selection Commands” on page 302 and “Clipboard Commands” on page 279).

Join Lines (Ctrl + P)

Joins two lines of text by replacing the new line character from the end of the current line and all initial whitespace on the next line with a single space.

This menu item is bound to the **JoinLines** command (see “Block Commands” on page 277).

Insert File

Opens the **Insert** dialog box, which allows you to select a file to be inserted. The contents of the inserted file are placed on the line above the cursor.

This menu item is bound to the **InsertFile** command (see “Insert Commands” on page 293).

The Tools Menu

The **Tools** menu contains the following items:

Insert Date (Ctrl + Shift + D)

Inserts the current date, as a formatted string, at the current location.

This menu item is bound to the **Date** command (see “Insert Commands” on page 293).

Search in Files

Opens the Search in Files dialog box, which allows you to search for an expression in all open files. See “Searching in Files” on page 33 for a description of this window and how to use it.

This menu item is bound to the **Grep** command (see “Tools Commands” on page 310).

Launcher

Opens the MULTI **Launcher**, which provides easy access to all the primary MULTI components. The Launcher provides a convenient way to create new files and projects, to access recent ones, and to manage your open windows.

This menu item is bound to the **MultiBar** command (see “Tools Commands” on page 310).

For more information about the Launcher, see Chapter 4, “Launcher GUI Reference”.

Make

UNIX only

Opens the **Thing to make?** dialog box. Changes made in the Editor are saved before **Make** starts. To examine erroneous lines, click error messages that appear in the output window.

Note: If you opened the Editor from the Debugger, selecting this menu item runs **make** on the executable you were debugging.

This menu item is bound to the **Make** command (see “Tools Commands” on page 310).

Execute Shell Command

UNIX only

Opens the **Shell Command to execute?** dialog box, which allows you to execute a command in the shell, then inserts the output into the open file.

The command uses the current selection in the Editor as `stdin`, and replaces that selection with `stdout`. If nothing is selected, the output from the command is inserted into the open file after the cursor's current position.

The following macro sequences are recognized in the *command*:

- `%FILE` — Replaced with the name of the current open file.
- `%SEL` — Replaced with the current selection.
- `%LINE` — Replaced with the current line number.
- `%COMMENTS` — Replaced with text from a dialog box that prompts for input.

This menu item is bound to the **ExecuteCmd** command (see “Tools Commands” on page 310).

Command to Window

UNIX only

Opens the **Command to Window** dialog box, which allows you to execute a command in the shell. Output will appear in a new temporary Editor window. This command does not modify the open file.

The following macro sequences are recognized in the *command*:

- `%FILE` — Replaced with the name of the current open file.
- `%SEL` — Replaced with the current selection.
- `%LINE` — Replaced with the current line number.
- `%COMMENTS` — Replaced with text from a dialog box that prompts for input.

This menu item is bound to the **CommandToWindow** command (see “Tools Commands” on page 310).

Notepad

Opens a small Editor window on a scratch file.

The scratch file used is:

- Windows — **multipad.txt** located in your **My Documents** directory.
- UNIX — **.Notes** located in your home directory.

This menu item is bound to the **Notepad** command (see “Tools Commands” on page 310).

Launch Utility Programs

Opens the **Utility Program Launcher** dialog box, which allows you to launch one of the Green Hills utility programs. For information about the utility programs, see the *MULTI: Building Applications* book for your target processor family.

This menu item is bound to the **WGUtils** command (see “Tools Commands” on page 310).

Execute Editor Commands

Opens the **Execute Editor Commands** dialog box, which allows you to enter Editor commands.

This menu item is bound to the **Minibuffer** command (see “Tools Commands” on page 310).

Append TagFile

Opens a dialog box where you can specify a tag file (in the format determined by the **ctag** utility) that the Editor can use to find procedures in files. By default, the Editor looks for a file called **tags**. For more information, see “Using Tags in Your Files” on page 27.

This menu item is bound to the **AppendTagFile** command (see “Tag Commands” on page 307).

Reset Tags

Reloads tags from the default tag file (**tags**) and from any additional tag files specified via the **Tools** → **Append TagFile** menu item (see above) or the **AppendTagFile** command (see “Tag Commands” on page 307).

This menu item is bound to the **ResetTags** command (see “Tag Commands” on page 307).

Merge Files

Opens the **Merge** dialog box, which allows you to merge two or three files. For more information, see “Merging Files” on page 40.

This menu item is bound to the **MergeFiles** command (“Tools Commands” on page 310).

Diff Files

Opens the **Diff Files** dialog box, which allows you to find and display the differences between two files. For more information, see “Comparing Files” on page 45.

This menu item is bound to the **DiffFiles** command (see “Tools Commands” on page 310).

The Version Menu

Most of the items in the **Version** menu are enabled only if the current file uses version control. The availability of the items listed in the table varies according to which version control system is being used. For more information, see Chapter 6, “Using Version Control with MULTI”.

Check Out

Retrieves a writable copy of the latest version and locks the file so that other users cannot change the file while you work on it. Checks out the current file from version control for editing. (Files cannot be edited unless they are checked out.) This action is not available if you are using CVS or Subversion.

This menu item is bound to the **CheckOut** command (see “Version Control Commands” on page 313).

Update

Issues an update command. If the repository version corresponding to the version of the local file has changed, the changes are merged into the local file. This menu item is only available for CVS and Subversion.

Check In/Commit

Saves any changes you have made to the current file. With some version control systems, the file becomes read only in MULTI, and the lock is removed from the file. You will be asked for comments to be saved in the log file along with your changes (see “The Commit Changes Dialog Box” on page 152). **Commit** is used for CVS and Subversion, and **Check In** is used for other version control systems.

This menu item is bound to the **CheckIn** command (see “Version Control Commands” on page 313).

Check In All

Saves the changes you have made to all the files you have checked out. With some version control systems, the files become read only in MULTI, and the lock is removed from the files. You will be asked for comments to be saved in the log file along with your changes.

This menu item is bound to the **QuerySaveCheckinAll** command (see “Version Control Commands” on page 313).

Discard Changes

Undoes all changes made to the file since it was checked out from version control, and reverts to the latest version. With some version control systems, the file becomes read only in MULTI, and the lock is removed from the file. This function updates the timestamp on a file in case the modified version was used in a previous build. Use this to undo a checkout without making any changes.

This menu item is bound to the **Discard** command (see “Version Control Commands” on page 313).

Place Under VC

Puts the current file under version control, prompting you to insert comments (if desired). With some version control systems, the file must be checked out before changes can be made.

This menu item is bound to the **PlaceUnderVC** command (see “Version Control Commands” on page 313).

Auto Checkout

Enables or disables Auto Checkout mode. If **Auto Checkout** is not selected, you must manually check out a file that uses version control before you can edit the file. If **Auto Checkout** is enabled, files will be automatically checked out from the version control system when you initiate an editing action.

This option functions on a per-file basis and defaults to the value of the **Automatic Checkout** configuration option. For more information, see Chapter 8, “Configuring and Customizing MULTI” .

This menu item is bound to the **AllowAutoCheckout** and **PreventAutoCheckout** commands (see “Version Control Commands” on page 313).

Show History

Opens the History Browser, which displays information about all versions of the current file. This window displays the check-in comment, who checked the file in, and the date of the check-in. You can double-click any entry in the history to open the Editor on a temporary copy of that version. See “The History Browser” on page 154 for more information. (If you are using ClearCase, this command opens the ClearCase history browser rather than the MULTI History Browser.)

This menu item is bound to the **ShowHistory** command (see “Version Control Commands” on page 313).

Show Last Edit

Finds the version of the current file that changed the selected text, or if no text is selected, the current line. A **Diff Viewer** window opens and displays the version that changed the text. (For more information about using this window, see “The Diff Viewer” on page 157.)

This menu item is bound to the **ShowLastEdit** command (see “Version Control Commands” on page 313).

Revert To History

Opens the History Browser, which lists all versions of the current file, and from which you can select a version to load into the Editor. (This is not supported for all version control systems.)

This menu item is bound to the **RevertHistory** command (see “Version Control Commands” on page 313).

Revert To Date

Opens a dialog box in which you can enter a revision date. The Editor will load the version that was current on the date you specify.

This menu item is bound to the **RevertDate** command (see “Version Control Commands” on page 313).

Revert To Version

Opens a dialog box in which you can enter a version number of the current file to load into the Editor. (This is not supported for all version control systems.)

This menu item is bound to the **RevertVersion** command (see “Version Control Commands” on page 313).

The Config Menu

The **Config** menu contains the following items:

Options

Opens the **Options** window to change options that affect the appearance and behavior of the Editor and other MULTI tools. This dialog box contains is covered in detail in Chapter 9, “Configuration Options”.

This menu item is bound to the **ConfigOptions** command (see “Configuration Commands” on page 282).

Customize Menus

Opens the **Customize Menus** window, which allows you to configure menus in a number of MULTI tools. For information about how to use this window, see “Configuring and Customizing Menus” on page 186.

This menu item is bound to the **CustomizeMenus** command (see “Configuration Commands” on page 282).

Save Configuration as User Default

Saves the current configuration options in a file in the default location. For more information, see “Saving to the User Configuration File” on page 172.

This menu item is bound to the **SaveConfig** command (see “Configuration Commands” on page 282).

Clear User Default Configuration

Prompts before clearing the default configuration file and restoring the default system configuration.

This menu item is bound to the **ClearConfig** command (see “Configuration Commands” on page 282).

Save Configuration As

Opens the **Save Configuration to what file?** dialog box, which allows you to specify the filename and location for the configuration file. For more information, see “Saving Configuration Settings” on page 172.

This menu item is bound to the **SaveConfigToFile** command (see “Configuration Commands” on page 282).

Load Configuration

Opens the **Load Configuration from what file?** dialog box, which allows you to specify the filename and location of the configuration file you want to load.

This menu item is bound to the **LoadConfigFromFile** command (see “Configuration Commands” on page 282).

The Windows Menu

The **Windows** menu is used to jump between all of the windows created by MULTI. If you choose an entry in the **Windows** menu, the corresponding window is brought to the front.

The following are menu items that may appear in the **Windows** menu:

debugging_window

Gives focus to a debugging-related window that has been launched from the local executable.

application_name

Gives focus to the main Debugger window for the local *application*.

IDE_tool

Opens a submenu that lists windows corresponding to the *IDE_tool*. The windows listed in this submenu might be created from the local execution or from other executions.

If only one MULTI Launcher is open, that tool does not have a submenu; it has a menu entry instead.

Misc

Opens a submenu listing other windows types that do not fall into any of the above categories.

The Help Menu

The **Help** menu contains the following items:

Editor Help (F1)

Opens MULTI's online help system.

This menu item is bound to the **Help** command (see "Help Commands" on page 289).

Manuals

Opens the **Manuals** submenu, which displays a list of manuals appropriate to your version of MULTI. Select one of these manuals to open the online help to the first page of that manual.

Bookmarks

Opens the **Bookmarks** submenu, which displays all the Help Viewer bookmarks you have created. Bookmarks function across manuals and MULTI sessions. Select a bookmark to display the bookmarked page in online help. Select **Manage bookmarks** to open a window that allows you to display bookmarked pages and rename, delete, or reorder bookmarks.

Identify

Waits until the next key or mouse click sequence, and displays help for that command instead of executing the command.

This menu item is bound to the **Identify** command (see "Help Commands" on page 289).

About MULTI

Opens the **About** banner.

This menu item is bound to the **About** command (see “Help Commands” on page 289).

License Info

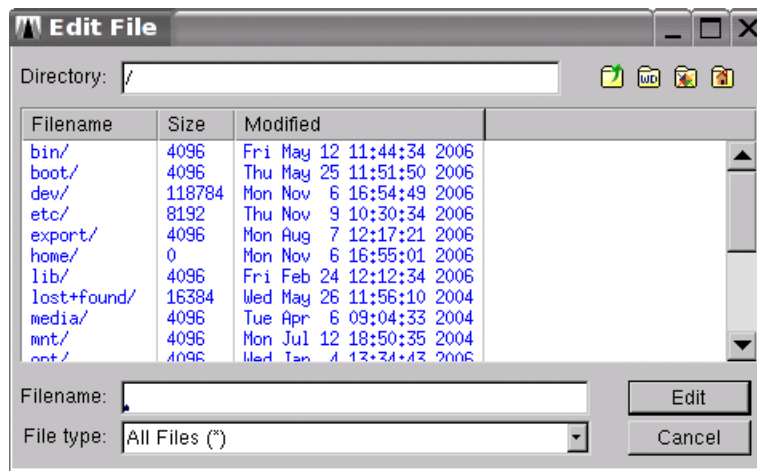
Opens the **Active Licenses** dialog box, which displays the licenses in use.

UNIX Dialog Boxes

The following sections describe dialog boxes that only appear in the Editor on UNIX systems.

The File Chooser Dialog Box (UNIX)

A file chooser dialog box opens if you initiate an action (using a menu, button, command, or shortcut) for which a file must be specified, but you have not yet specified a file. A chooser also opens if you click a **Browse** or  button on a dialog box. A sample UNIX file chooser follows. (For information about Windows file choosers, see your Windows documentation.)



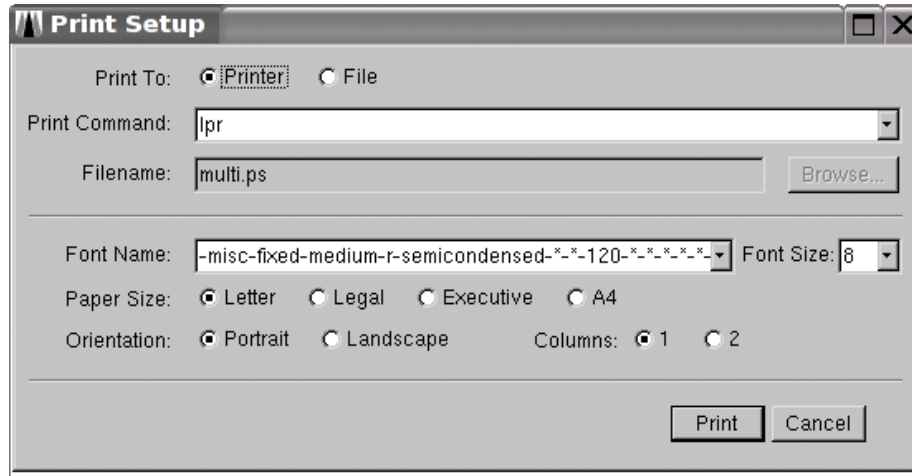
The title bar of the file chooser will vary depending on the action you are performing. The following table describes the main elements of the file chooser.

Item	Meaning
Directory	Displays the directory whose contents are shown in the File List . Type in a new directory name and press Enter to display a different directory.
Directory Buttons	With this set of buttons, you can quickly go to different important directories. The buttons that appear are:  — Up One Directory from the current directory.  — Jumps to the current Working Directory.  — Jumps to the MULTI Run Directory.  — Jumps to the User Home Directory.
File List	Below the directory text field is the file list. To enter a directory, double-click the directory. To choose a file, single-click the filename. You can click any column header to sort the list in ascending or descending order. If multiple files are allowed for the present operation (for example, Open is selected in the editor menu), use the Shift key to select a consecutive list of files; use the Ctrl key to select non-consecutive files.
Filename	Type a filename or directory name into this text field. As you type in this field, the selection in the file list will change to reflect the closest match. If you type a directory name and press Enter, or follow the directory name by a slash (/), the file list will change to the specified directory.
File type	If you select a file type, the File List will only display files with suffixes that match the selected file type.
Action buttons	There are two buttons in the lower right corner of the file chooser window. The upper button displays the action that will occur, such as Open (in the example above), OK , or Edit . The lower button is the Cancel button, which closes the window without taking any action.

The Print Setup Dialog Box

On Windows, selecting a printing operation from the **File** menu opens a standard **Print** dialog box with settings and options appropriate for your version of Windows and your printer and printer driver. See your Windows and printer documentation for details about the settings on this dialog.

On UNIX systems, selecting a printing operation from the **File** menu opens the following **Print Setup** dialog box.



The settings of the **Print Setup** dialog box are described in the following table.

Item	Meaning
Print To	Specifies where to send the print output. Select either Printer (the default) or File (to write to a postscript file).
Print Command	Specifies the print command that will be run when you click Print. Enter the appropriate command and options for printing a file on your specific system (for example, lpr on UNIX). You can also set the print command with the configure printCommand command. For information about the configure command, see “Using the configure Command” in Chapter 8, “Configuring and Customizing MULTI” in the <i>MULTI: Editing Files and Configuring the IDE</i> book. This field is only available if you have chosen the Printer radio button above.
Filename	Specifies the output post-script file for print to file operations. This field is only available if you have selected the File radio button above. Enter the file to write to, or click Browse to open a chooser and navigate to an existing file. (See “The File Chooser Dialog Box (UNIX)” on page 76 for a description of the chooser.)
Font Name	Specifies the font for your printing operations. Select your desired font from the drop-down list, which contains a list of available fonts on your system. You can also type a font name into this field if it is not in the list.

Item	Meaning
Font Size	Specifies the font size used for printing. Select your desired size from the drop-down list. The default font size is 8 point.
Paper Size	Specifies your paper size. Select Letter , Legal , Executive , or A4 . The default setting is Letter .
Orientation	Specifies the layout for your printed document. Select Portrait or Landscape . The default setting is Portrait .
Columns	Specifies whether to print the output in one or two columns. The default setting is 1 .
Print	Executes the print request.
Cancel	Cancels the print request and discards any settings you have changed.

Managing the MULTI IDE

Part II

Launcher GUI Reference

Chapter 4

This Chapter Contains:

- Starting the Launcher
- The Launcher Menus
- The Launcher Toolbar
- The Detail Pane

The MULTI Launcher is the primary entry point to the MULTI IDE. It provides an easy way to launch all the major IDE components and manage user-defined workspaces and shortcuts. This chapter provides detailed information about the Launcher GUI.

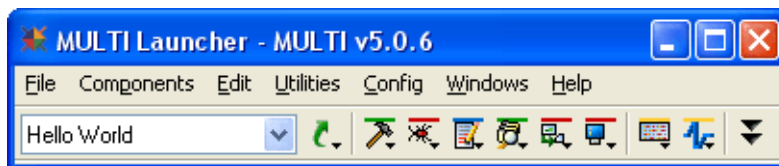
Starting the Launcher

To start the Launcher:

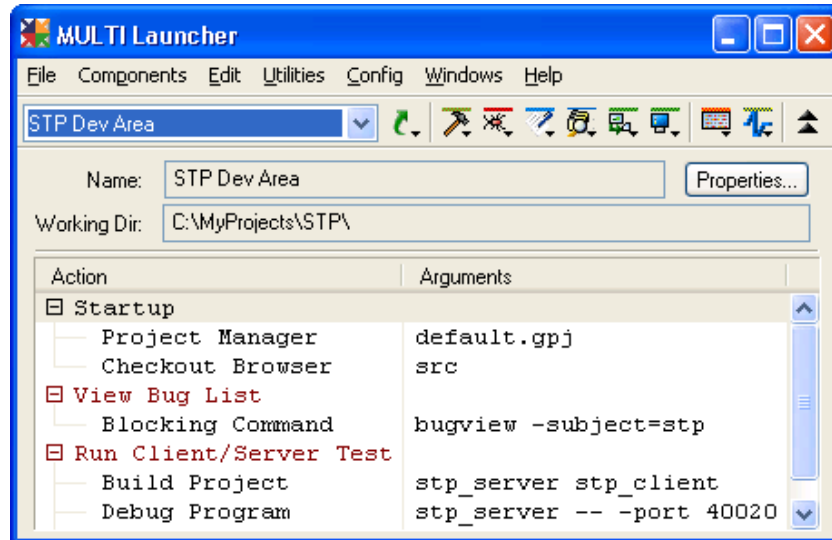
- Windows — From the Windows **Start** menu, select the MULTI menu item. Alternatively, use Windows Explorer to navigate to the directory where you installed MULTI, and double-click **multi.exe**.
- UNIX — Ensure the MULTI installation directory is in your path and run **multi**.

The Launcher has two display modes.

Concise mode (shown below) uses little screen space. This mode is useful if you do not need to access or modify individual actions. For information about modifying actions, see “Creating or Modifying an Action” on page 109.



Detailed mode (shown below) shows more information and allows you to modify workspace action sequences. For more information about this mode, see “The Detail Pane” on page 97. For information about modifying workspace action sequences, see “Creating or Modifying an Action Sequence” on page 107.



You can switch between the two modes by clicking the  or  button. The current mode is maintained across Launcher sessions.

The Launcher Menus

The following sections describe the menu items available in the Launcher. Some menu items are dimmed out if they are unavailable in your current context.

The Launcher menus are:

- The **File** menu (page 86)
- The **Components** menu (page 87)
- The **Edit** menu (page 88)
- The **Utilities** menu (page 90)
- The **Config** menu (page 91)
- The **Windows** menu (page 92)
- The **Help** menu (page 93)

The File Menu

The **File** menu contains the following menu items:

Create Workspace
Opens the Select Workspace Type dialog box, which you can use to create a new workspace. For more information, see “Creating Workspaces” on page 102.
Delete Workspaces
Opens the Delete Workspaces dialog box, which allows you to select one or more workspaces for deletion.
Recent Workspaces
Opens a submenu listing the most recently accessed workspaces. If you select one, it becomes the current workspace.
Load Workspace from File
Opens a dialog box that allows you to import a workspace from a .gmb file. For more information, see “Importing and Exporting Workspaces” on page 106.
Load Shortcuts from File
Opens a dialog box that allows you to import shortcuts from a .gmb file. For more information, see “Importing and Exporting Shortcuts” on page 114.
Create or Change Shortcuts
Allows you to create a new shortcut or change an existing shortcut. For general information about shortcuts, see “Overview of Workspaces and Shortcuts” on page 100. For more information about creating shortcuts, see “Creating MULTI Shortcuts” on page 113.
Create New Project
Opens the Project Wizard , which helps you create a new project. After you create the project, it loads in the Project Manager. For more information, see Chapter 2, “The MULTI Project Manager” in the <i>MULTI: Building Applications</i> book for your target processor family.
Save Changes
Saves all workspace, shortcut, and Launcher property changes. The changes are saved into the index.gmb file. For more information, see “Saving Workspaces” on page 106 and “Saving Shortcuts” on page 114.
Save Current Workspace into File
Opens a dialog box that allows you to export the current workspace to a specified file. For more information, see “Importing and Exporting Workspaces” on page 106.

Save Shortcuts into File

Opens a dialog box that allows you to export all of your shortcuts to a specified file. For more information, see “Importing and Exporting Shortcuts” on page 114.

Close Launcher

Closes the Launcher. Any unsaved changes to workspaces or Launcher properties are saved automatically.

Exit All

Closes all MULTI windows, including the Launcher.

The Components Menu

The **Components** menu contains the following menu items:

Open Project Manager

Launches an empty Project Manager.

Open Debugger

Launches a file chooser that allows you to select a program to debug. The Debugger opens on the program you select.

Open Editor

Launches a file chooser that allows you to select a file to edit. The Editor opens on the file you select.

Open Diff Viewer

Opens a window in which you can specify two files for comparison. The files can either be two different files or two different versions of the same file. For more information, see “The Diff Viewer” on page 157.

Open Checkout Browser

Launches the Checkout Browser from the current working directory.

Connect

Launches the **Connection Chooser** so that you can establish a connection.

Open Connection Organizer

Launches the **Connection Organizer** so that you can manipulate your connections.

Open Serial Terminal

Launches the Serial Terminal.

Open EventAnalyzer

Launches the MULTI EventAnalyzer.

This entry will not be present if the MULTI EventAnalyzer is not licensed or if the executable does not exist in your distribution.

Open ResourceAnalyzer

Launches the MULTI ResourceAnalyzer.

This entry will not be present if the MULTI ResourceAnalyzer is not licensed or if the executable does not exist in your distribution.

Open Python GUI

Opens the stand-alone **MULTI Python GUI** window.

The Edit Menu

The **Edit** menu contains the following menu items:

Add Action Sequence

Adds a new action sequence to the current workspace. For information about action sequences, see “Overview of Workspaces and Shortcuts” on page 100 and “Action Sequences and Actions” on page 107.

Add Action (Ctrl + A)

Opens the **Edit Action** dialog box, which allows you to add an action to the end of the current action sequence.

For more information about the **Edit Action** dialog box, see “Creating or Modifying an Action” on page 109. For information about actions, see “Overview of Workspaces and Shortcuts” on page 100.

Edit Selected Object (Ctrl + E)

Opens the **Edit Action Sequence** dialog box if an action sequence is selected. Opens the **Edit Action** dialog box if an action is selected. These dialog boxes allow you to modify the properties of the action or action sequence.

For more information about the **Edit Action Sequence** dialog box, see “Creating or Modifying an Action Sequence” on page 107. For more information about the **Edit Action** dialog box, see “Creating or Modifying an Action” on page 109.

Cut Selected Actions (Ctrl + X)

Cuts the selected action(s) and places them in the internal buffer.

Copy Selected Actions (Ctrl + C)

Copies the selected action(s) into an internal buffer.

Paste Actions (Ctrl + V)

Pastes the actions in the internal buffer before the selected action or at the end of the selected action sequence.

Delete Selected Objects (Ctrl + D or Delete)

Deletes the selected actions and/or action sequences.

Move Selected Objects Up (Ctrl + UpArrow)

Moves the selected action sequence up by one level or the selected action up by one row (see the note below).

Move Selected Objects Down (Ctrl + DownArrow)

Moves the selected action sequence down by one level or the selected action down by one row (see the note below).

Run Selected Objects (Ctrl + R)

Runs the selected actions and/or action sequences (see the note below).



Note When you apply a **Move Selected Objects Up**, **Move Selected Objects Down**, or **Run Selected Objects** operation to selected objects in the action tree (either from the **Edit** menu, the right-click shortcut menu, or with a keyboard shortcut), the operation is applied as follows:

- If any actions are selected, the operation is applied only to those selected actions and not to entire action sequences.
- Otherwise, the operation is applied to the selected action sequences.

When you apply the **Run Selected Objects** operation to an object:

- If the object is an action, the action executes even if disabled.
- If the object is an action sequence, only enabled actions in the action sequence execute.

The Utilities Menu

The **Utilities** menu contains the following entries:

Running Actions
Lists running processes that have been launched by workspace actions. Select a process to terminate it. This submenu does not list components of the MULTI IDE that were launched by Launcher workspace actions. For more information, see “Managing Running Actions” on page 112.
Clean Last Action Execution
Undoes the last high-level operation. For example, selecting this menu item may exit the last MULTI IDE tool you opened, along with closing that tool’s associated windows, or it may disconnect from the last target you connected to. Select this menu item once for each operation you want to undo.
License Administrator
Opens the MULTI Licensing Wizard . For more information, see Chapter 2, “The MULTI License Administrator” in the <i>MULTI: Licensing</i> book.
Probe Administrator
Launches the Green Hills Probe Administrator to manage the Green Hills probes on your local network. This entry will not be present if the Green Hills Probe Administrator is not licensed or if the executable does not exist in your distribution
Launch Utility Programs
Opens the Utility Program Launcher dialog box, which allows you to launch one of the Green Hills utility programs. For information about the utility programs, see the <i>MULTI: Building Applications</i> book for your target processor family.

The Config Menu

The **Config** menu contains the following entries:

Show/Hide Detail Pane

Toggles whether the detail pane is visible in the Launcher. For more information about the detail pane, see “The Detail Pane” on page 97.

Start with MULTI Launcher

Specifies the effect of entering the **multi** command on the command line.

- If a check mark appears next to this menu item, issuing the **multi** command with no arguments runs the Launcher. This is the default behavior.
- If no check mark appears next to this menu item, issuing the **multi** command with no arguments runs the Project Manager.

To toggle between these settings, click the menu item.

Show Progress Window

Toggles the display of a MULTI Editor progress window that prints output for running workspace actions. For more information about the progress window and when it appears, see “Managing Running Actions” on page 112.

Abort Execution on Error

Toggles whether or not the Launcher stops execution of multiple actions when one of the following errors is encountered in an action:

- The Launcher cannot communicate with the specified MULTI IDE component.
- For **Project Manager** actions, the specified file is not a project file.
- For **Debug Program** actions, the specified program does not exist.

For example, if you select and run three sequential actions, and one of these errors is encountered in the second action, the third action will not execute if this menu item is enabled.

For information about actions, see Chapter 5, “Using MULTI Workspaces and Shortcuts”.

Options

Opens the **Options** window. For information about the **Options** window, see Chapter 9, “Configuration Options”.

Customize Menus

Opens the **Customize Menus** window, which allows you to configure menus in a number of MULTI tools. For information about how to use this window, see “Configuring and Customizing Menus” on page 186.

Save Configuration as User Default Saves the current configuration options in the default location. For more information, see “Saving to the User Configuration File” on page 172.
Clear User Default Configuration Clears the default configuration file and restores the default system configuration. The Launcher prompts you before performing this action.
Save Configuration As Opens the Save Configuration to what file? dialog box, which allows you to specify the filename and location in which to save the configuration. For more information, see “Saving Configuration Settings” on page 172.
Load Configuration Opens the Load Configuration from what file? dialog box, which allows you to specify the filename and location of the configuration file to load.
Set INTEGRITY Distribution Opens the Default INTEGRITY Distribution dialog box, where you can set the location of your INTEGRITY installation directory. For more information, see “Using MULTI with INTEGRITY or u-velOSity” in Chapter 2, “Installation” in the <i>MULTI: Getting Started</i> book.
Set u-velOSity Distribution Opens the Default u-velOSity Distribution dialog box, where you can set the location of your u-velOSity installation directory. For more information, see “Using MULTI with INTEGRITY or u-velOSity” in Chapter 2, “Installation” in the <i>MULTI: Getting Started</i> book.

The Windows Menu

The **Windows** menu lists all open windows in the MULTI IDE. Select an item from the menu to bring the corresponding window to the top.

The **Windows** menu may contain the following entries:

<i>IDE_tool</i> Opens a submenu that lists windows corresponding to the <i>IDE_tool</i> . If there are no other MULTI windows in the system, this menu item is not displayed.
Misc Lists all other open MULTI windows in the system. If there are no other MULTI windows in the system, this menu item is not displayed.

Minimize All Windows

Minimizes all MULTI windows except the Launcher.

If there are no MULTI windows open, this menu item is not displayed.

Show All Windows

Restores all MULTI windows to their normal sizes.

If there are no MULTI windows open, this menu item is not displayed.

Close All Windows

Closes all MULTI windows. Before certain windows close, you may be prompted. For example:

- Before a Debugger is closed, you may be asked whether to kill the process being debugged.
- Before an Editor is closed, you may be asked whether to save outstanding changes.

If there are no MULTI windows open, this menu item is not displayed.

No Windows

Indicates that no MULTI windows (except for the Launcher itself) are open.

The Help Menu

The **Help** menu contains the following entries:

Show Information

Shows the Launcher information in the Detail Pane.

This information includes a brief overview of how to use the Launcher.

Help

Opens MULTI's online help system.

Manuals

Opens the **Manuals** submenu, which displays a list of manuals appropriate to your version of MULTI. Select one of these manuals to open the online help to the first page of that manual.

Bookmarks

Opens the **Bookmarks** submenu, which displays all Help Viewer bookmarks you have created. Bookmarks function across manuals and MULTI sessions. Select a bookmark to display the bookmarked page in online help. Select **Manage bookmarks** to open a window that allows you to display bookmarked pages and rename, delete, or reorder bookmarks.

About MULTI Launcher
Shows basic information about MULTI, including the version number and revision date.
License Info
Opens the Active Licenses dialog box, which displays the licenses in use.
Troubleshooting Info
Gathers information about your MULTI distribution, including your system information, environment information and your MULTI installation information, and displays it in a dialog box. You can save the information to a file or send it to others, such as Green Hills support or your product support contact.

The Launcher Toolbar

The Launcher toolbar contains buttons that allow you to access workspaces, shortcuts, and major MULTI IDE components.

The following list gives a full description of each Launcher item:

- *Workspace* drop-down menu — Displays the current workspace. You can view the full list of available workspaces by clicking the drop-down button. Imported workspaces are indicated by .

To change the current workspace, select a different workspace from the drop-down. For information about workspaces, see “Overview of Workspaces and Shortcuts” on page 100.

 **(Run Action Sequences or Shortcuts)** — Lists the following items:

- *Action sequences* — Executes the selected action sequence. Action sequences appear in the order they are defined in the current workspace.
- *Shortcuts* — Executes the selected shortcut. Shortcuts appear in the order they were used, with the most recently used shortcut appearing first.
- **Create Workspace** — Opens a window in which you can create a new workspace. For more information, see “Creating Workspaces” on page 102.
- **Create or Change Shortcut** — Allows you to create or modify shortcuts just as you would action sequences for a workspace. For more information, see “Creating MULTI Shortcuts” on page 113.

 **(Launch Project Manager)** — Lists the following items:

- *Projects* — Opens the selected project in the Project Manager. If the current workspace uses projects as arguments to **Project Manager** actions, these projects are listed first. Recently used projects are listed next.
- **Open Project Manager** — Launches an empty Project Manager.
- **Create Project** — Opens the **Project Wizard**, which helps you create a new project. After you create the project, it loads in the Project Manager.

 **(Launch Debugger)** — Lists the following items:

- *Programs* — Opens the selected program in the Debugger. If the current workspace uses programs as arguments to **Debug Program** actions, these programs are listed first. Recently used programs are listed next.
- **Open Debugger** — Launches a file chooser that allows you to select a program to debug. The Debugger opens on the program you select.

 **(Launch Editor)** — Lists the following items:

- *Files* — Opens the selected file in the Editor. If the current workspace uses files as arguments to **Editor** actions, these files are listed first. Recently used Editor files are listed next.
- **Open Editor** — Launches a file chooser that allows you to select a file to edit. The Editor opens on the file you select.

 **(Launch Checkout Browser)** — Lists the following items:

- *Directories* — Opens a Checkout Browser on the selected directory. If the current workspace uses directories as arguments to **Checkout Browser** actions, these directories are listed first. Recently used Checkout Browser directories are listed next.
- **Open Checkout Browser** — Launches the Checkout Browser from the current working directory.

 **(Connect to Target)** — Lists the following items:

- *Connection Methods* — Establishes a connection to your target. If the current workspace uses Connection Methods as arguments to **Connection** actions, these Connection Methods are listed first. Recently used Connection Methods are listed next.

- **Disconnect:** *target* — Terminates the specified system connection.
- **Connect** — Launches the **Connection Chooser**, which you can use to establish a connection.
- **Open Connection Organizer** — Launches the **Connection Organizer**, which you can use to manipulate your connections.



(Connect to Serial Terminal) — Lists the following items:

- **Serial Terminal Targets** — Establishes a serial terminal connection to the selected serial target. If the current workspace uses serial terminal targets as arguments to **Serial Terminal** actions, these serial targets are listed first. Recently used serial terminal targets are listed next.
- **Open Terminal** — Launches the Serial Terminal.



(Launch EventAnalyzer) — This button will not be present if the MULTI EventAnalyzer is not licensed or if the executable does not exist in your distribution. Lists the following items:

- **Event Stream Files** — Opens the selected event stream file in the MULTI EventAnalyzer. If the current workspace uses event stream files as arguments to **EventAnalyzer** actions, these files are listed first. Recently used event stream files are listed next.
- **Open EventAnalyzer** — Launches the MULTI EventAnalyzer.



(Launch ResourceAnalyzer) — This button will not be present if the MULTI ResourceAnalyzer is not licensed or if the executable does not exist in your distribution. Lists the following items:

- **ResourceAnalyzer Targets** — Launches the MULTI ResourceAnalyzer, which tries to connect to the selected target. If the current workspace uses ResourceAnalyzer targets as arguments to **ResourceAnalyzer** actions, these targets are listed first. Recently used ResourceAnalyzer targets are listed next.
- **Open ResourceAnalyzer** — Launches the MULTI ResourceAnalyzer GUI.



(Quit) — Closes the Launcher.



(Show Detail Pane) or  **(Hide Detail Pane)** — Expands or shrinks the Launcher window to show or hide the detail pane.

The Detail Pane

The detail pane is located below the toolbar and is displayed when the Launcher is in detailed mode. To see what detailed mode display looks like, refer to “Starting the Launcher” on page 84.

The detail pane displays either brief help information about the Launcher or information for the current workspace. When you create a new workspace or select a workspace in detailed mode, the detail pane displays the workspace’s information. For information about workspaces, see “Overview of Workspaces and Shortcuts” on page 100.

When displaying help information, the detail pane contains the following elements:

- **Help information pane** — Displays information about the Launcher.
- **Don’t show this message again** — Select this check box to hide help information when the Launcher next starts up.
- **Start with MULTI Launcher** — Select this check box to start the Launcher when the **multi** command is run with no arguments. See the corresponding menu entry in “The Config Menu” on page 91.

When displaying information for the current workspace (see “Overview of Workspaces and Shortcuts” on page 100), the detail pane contains the following elements:

- **Name** — Shows the workspace name.
- **Working Dir** — Shows the workspace working directory.
- **Properties** — Opens the **Set Workspace Properties** dialog box. The **Current** tab allows you to change the current workspace’s basic information, and the **Global** tab allows you to change certain global settings. Both tabs also list variables and allow you to create, edit, and delete variables. The **Others** tab lists pre-defined MULTI Launcher variables. For more information about variables, see “Using Variables in the MULTI Launcher” on page 115.
- **Action tree** — Lists the action sequences and actions in the current workspace. For a list of menu items and keyboard commands that you can use to manipulate the action tree, see “The Edit Menu” on page 88. For more information about workspaces, action sequences, and actions, see Chapter 5, “Using MULTI Workspaces and Shortcuts”.

Using MULTI Workspaces and Shortcuts

Chapter 5

This Chapter Contains:

- Overview of Workspaces and Shortcuts
- MULTI Workspaces
- Action Sequences and Actions
- MULTI Shortcuts
- Using Variables in the MULTI Launcher

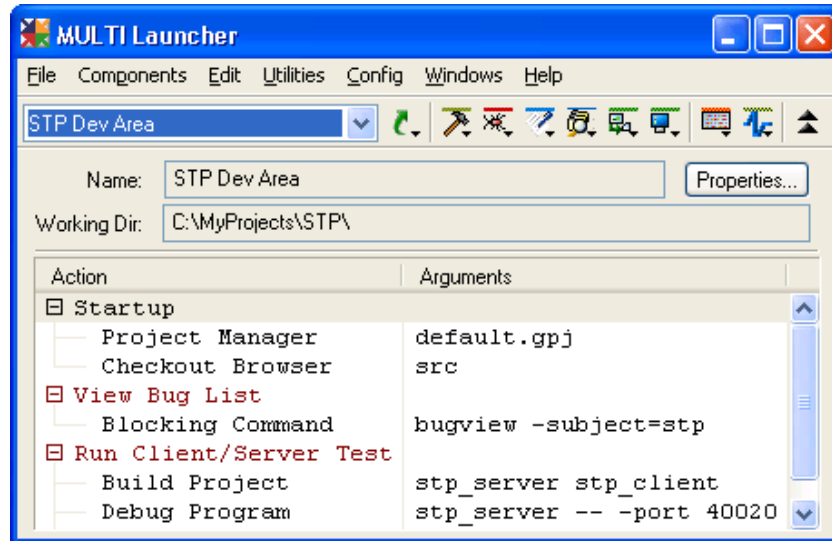
The MULTI Launcher allows you to create and use workspaces and shortcuts. A *workspace* is a virtual area where the tools, files, and actions required for a particular project can be organized, accessed, and executed. A *shortcut* is an action sequence that is not associated with a particular workspace. This chapter describes how to create, modify, and use workspaces and shortcuts.

Overview of Workspaces and Shortcuts

A workspace is typically created for each top-level project and includes a working directory and one or more related *action sequences*. The working directory for the workspace is the directory from which the action sequences will be started, and it is typically the directory containing the executable program associated with the project. An *action sequence* consists of one or more *actions* that are performed in the listed order. Action sequences may include:

- Building a program and then debugging it.
- Launching the Checkout Browser and opening several frequently used files for editing.
- Connecting to a Debugger hardware target and opening a serial terminal connection.
- Running a script or batch file or executing a Debugger command.
- Invoking an external tool.

A shortcut comprises an action sequence that is not associated with any particular workspace. Shortcuts can be run from within any workspace. They execute in the working directory of the currently selected workspace.



The example above displays a workspace called **STP Dev Area** with a working directory of **C:\MyProjects\STP** and action sequences named **Startup**, **View Bug List**, and **Run Client/Server Test**. The **Startup** action sequence contains the following actions:

- Open the MULTI Project Manager on the **default.gpj** project.
- Open the **Checkout Browser** on the **src** directory.

The **Run Client/Server Test** action sequence contains the following actions:

- Build the programs **stp_server** and **stp_client** contained in the **default.gpj** project.
- Open the MULTI Debugger on **stp_server** with arguments set to **-port 40020**.

The following sections provide more information about workspaces, actions, and shortcuts.

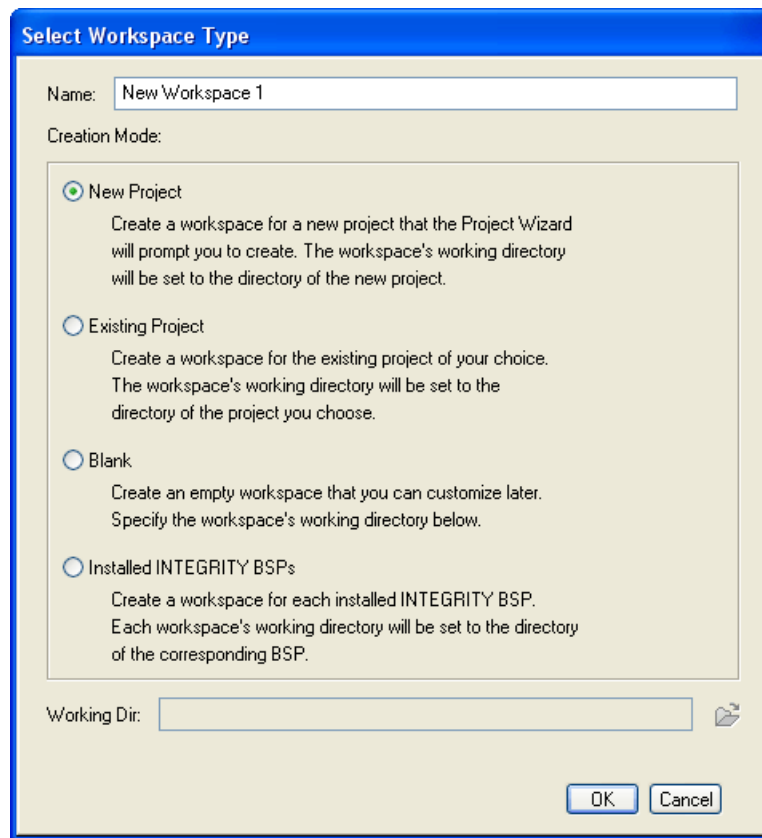
MULTI Workspaces

This section contains detailed information about creating, modifying, saving, loading, and opening MULTI workspaces. For general information about workspaces, action sequences, and actions, and to read through a workspace example, see “Overview of Workspaces and Shortcuts” on page 100.

Creating Workspaces

To create a workspace:

1. From the Launcher, select **File → Create Workspace**. The **Select Workspace Type** window appears.



2. Specify a name for your workspace and choose one of the following options:
 - **New Project** — Select this option to create a workspace for a new project. The **Project Wizard** opens to allow you to create the project.

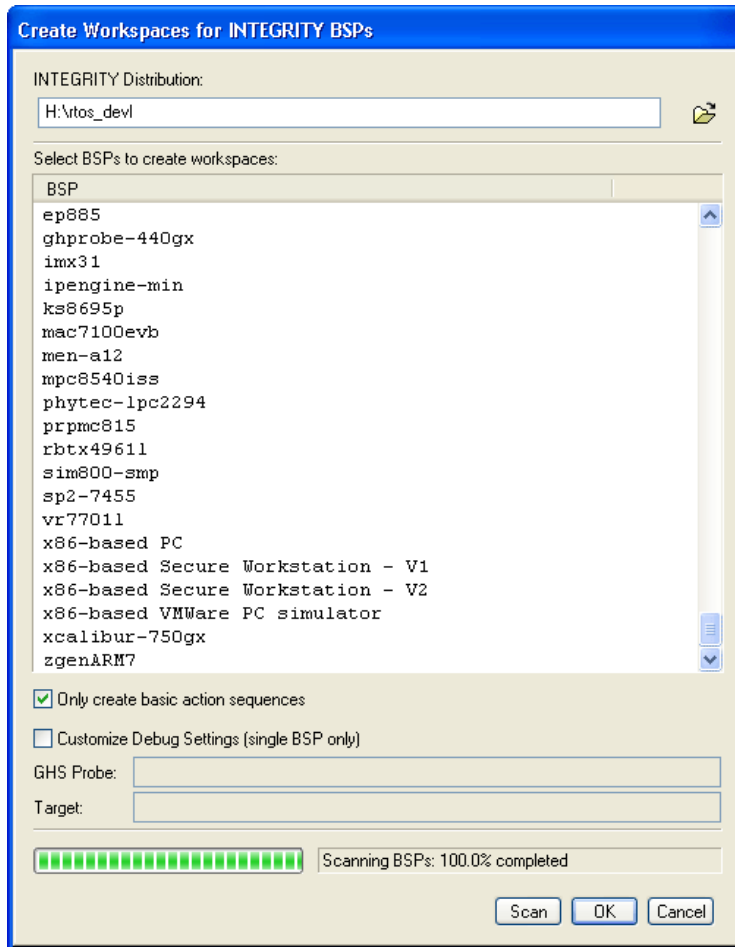
For more information, see “Creating a New Project” in Chapter 2, “The MULTI Project Manager” in the *MULTI: Building Applications* book.

- **Existing Project** — Select this option if you already have a top-level project and you want to create a workspace to provide you with easier access to that project. In the file chooser that appears, specify the location of your existing top-level project.
- **Blank** — Select this option to create an empty workspace (one that does not contain any action sequences). You can customize the workspace later. Enter the path to the desired working directory in the **Working Dir** field or click  to navigate to the desired working directory. For information about using variables to specify the working directory, see “Using Variables in the MULTI Launcher” on page 115.
- **Installed INTEGRITY BSPs** — Select this option to create one or more workspaces, each for an installed INTEGRITY BSP. Each workspace’s working directory is set to the directory of the corresponding INTEGRITY BSP. For information about the dialog box that appears when you choose this option, see “Creating an INTEGRITY BSP Workspace” on page 103.

When you create a workspace for a new project or an existing project, an action sequence called **Startup** is created automatically. The **Startup** action sequence launches the Project Manager on your project. “Creating or Modifying an Action Sequence” on page 107 explains how to customize action sequences and how to add other action sequences to your workspace.

Creating an INTEGRITY BSP Workspace

When you choose to create a workspace of type **Installed INTEGRITY BSPs** (see “Creating Workspaces” on page 102), the **Create Workspaces for INTEGRITY BSPs** dialog box appears.



The fields and options available in this dialog box allow you to customize your BSP workspace. Each field, option, and button is described in detail below.

- **INTEGRITY Distribution** — Specifies the INTEGRITY distribution directory to scan for BSPs. If the default directory is incorrect or if you want to choose among multiple INTEGRITY installations, click the  button located to the right of the field.
- **Select BSPs to create workspaces** — Lists the BSPs available in the **INTEGRITY Distribution** directory. Select one or more BSPs to create your workspace(s).
- **Only create basic action sequences** — Populates your workspace(s) with basic action sequences. If unselected, this option populates your workspace(s) with basic and advanced action sequences. This option is only applicable for simulator BSPs.

- **Customize Debug Settings (single BSP only)** — Creates connection actions for the specified **GHS Probe** and **Target**. These actions are used as part of the action sequence that downloads the kernel via the Probe and then launches a run-time debug connection on the specified target. This option is only applicable if you select a single hardware BSP, and you complete the **GHS Probe** and **Target** fields.
- **GHS Probe** — Specifies the name or IP address of your GHS Probe. This field is used to create connection actions and is only available when **Customize Debug Settings (single BSP only)** is selected.
- **Target** — Specifies the name or IP address of your target. This field is used to create a connection action and is only available when **Customize Debug Settings (single BSP only)** is selected.
- **Progress Bar** — Indicates the percentage of the BSP scan completed. Press Esc to abort the scan.
- **Scan** — Scans the specified directory for BSPs. The directory is specified in the **INTEGRITY Distribution** text field located at the top of the dialog box.
- **OK** — Creates the workspace(s).
- **Cancel** — Closes the dialog box and cancels workspace creation.

Changing Workspace Properties and Global Settings

After you have created a workspace, you can edit the **Working Dir** path and other basic workspace information by clicking the **Properties** button in the Launcher. If this button is not visible, click  to reveal the detail pane.

In the **Set Workspace Properties** dialog box, the **Current** tab allows you to change the current workspace's basic information and the **Global** tab allows you to change certain global settings. The **Others** tab lists pre-defined MULTI Launcher variables. For information about using variables in this dialog box, see "Using Variables in the MULTI Launcher" on page 115.

Saving Workspaces

To save your workspace to the default location, select **File → Save Changes** from the Launcher. The Launcher saves definitions of workspaces into the configuration file for the current workspace and into **index.gmb**, which is located at:

- Windows — *user_dir*\Application Data\GHS\launcher
- UNIX — *user_dir*/.ghs/launcher

To find the full path to **index.gmb**, make a change to your current workspace and select **File → Save Changes** from the Launcher. The dialog box that opens displays the full path.

To save your workspace to a different file, see “Importing and Exporting Workspaces” on page 106.

Importing and Exporting Workspaces

If you want to export a workspace, select **File → Save Current Workspace into File** from the Launcher. In the file chooser that opens, select the file in which you want to save your workspace. You can share this file with other users.

To import a workspace, select **File → Load Workspace from File** and choose a **.gmb** file. MULTI prompts you to choose whether you want to access the workspace directly from the imported file or copy the workspace to your local configuration and access the copied workspace. In the workspace drop-down menu, workspaces that are not copied to your local configuration are indicated by .

The workspace you import becomes the current workspace.

To add imported workspaces to a project, open the MULTI Project Manager on the project and select **Edit → Add File into default.gpj**.

Opening Workspaces

You can create as many workspaces as necessary, but only one workspace can be the *current* workspace, and you can only run actions from the current workspace. You can change the current workspace in either of the following ways:

- From the Launcher, open the workspace drop-down menu and select the desired workspace (see “The Launcher Toolbar” on page 94).
- From the Launcher, select **File** → **Recent Workspaces** and select the desired workspace.

Action Sequences and Actions

This section contains detailed information about:

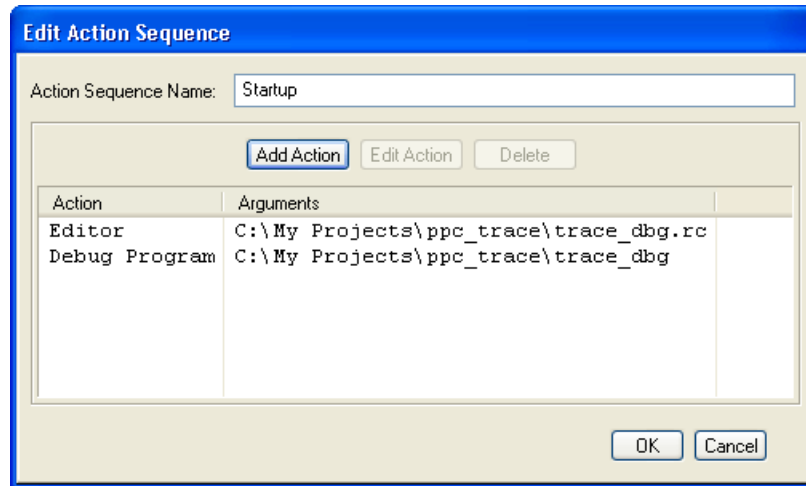
- Creating, modifying, and running actions and action sequences.
- Managing running actions.

For general information about actions and action sequences and to see how they are used in an example workspace, refer to “Overview of Workspaces and Shortcuts” on page 100. For information about the automatically created **Startup** action sequence, see “Creating Workspaces” on page 102.

Creating or Modifying an Action Sequence

To create an action sequence:

1. Select **Edit** → **Add Action Sequence** from the Launcher. This opens the **Edit Action Sequence** dialog box on a new action sequence.



To move the action sequence up or down in the action tree, right-click the action sequence and select **Move Up** or **Move Down**.

2. To add actions to your action sequence, follow the instructions in “Creating or Modifying an Action” on page 109.

To edit an action sequence, do one of the following:

- Select the action sequence and then select **Edit** → **Edit Selected Object**.
- Right-click the action sequence and select **Edit** from the shortcut menu.

Action sequences are modified in the **Edit Action Sequence** dialog box. You can use the buttons in this dialog box to add a new action to the action sequence, edit an existing action, or delete an action. You can also perform the following operations by right-clicking an action in this dialog box.

Edit

Displays the **Edit Action** dialog. For more information, see “Creating or Modifying an Action” on page 109.

Disable or Enable

Disables or enables the selected action. Disabled actions appear dimmed and do not run when the action sequence is run.

Cut, Copy, Paste, or Delete

Performs the requested editing operation on the selected action. The **Paste** button inserts actions above the current selection; it does not overwrite the current selection.

Move Up or Move Down

Moves the selected action or actions one position up (earlier) or down (later) in the list of actions.

To reorder an action sequence, select it and then do one of the following:

- Press **Ctrl + UpArrow** or **Ctrl + DownArrow** to move the action sequence up or down.
- Select **Edit → Move Selected Objects Up** or **Edit → Move Selected Objects Down**.

To delete an action sequence, do one of the following:

- Select the action sequence and press **Delete**.
- Right-click the action sequence and select **Delete** from the shortcut menu.

Creating or Modifying an Action

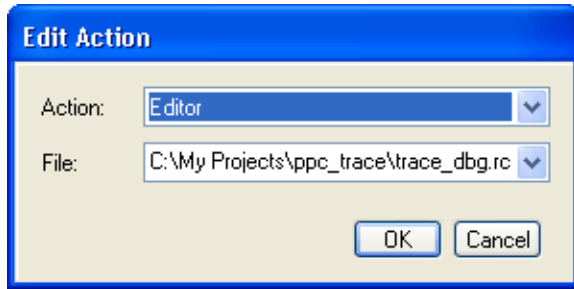
To create an action, perform one of the following operations:

- From the Launcher, right-click the action sequence and select **Add Action**.
- From the **Edit Action Sequence** dialog box, click **Add Action**.

To modify an action, perform one of the following operations:

- From the Launcher, right-click the action and choose **Edit**.
- From the Launcher, select the action and then choose **Edit → Edit Selected Object**.
- From the **Edit Action Sequence** dialog box, either right-click the action and choose **Edit**, or select the action and click **Edit Action**.

The **Edit Action** dialog box appears when you create or modify an action.



This dialog box contains two fields. The first field, **Action**, specifies the type of action. The second field (whose name is dependent on the **Action** selected) contains the arguments for the action. First select the type of action from the **Action** list, and then fill in the second field, which may contain variables. For information about using variables, see “Using Variables in the MULTI Launcher” on page 115.

Available action types and example arguments follow:

- **Project Manager** — Opens the MULTI Project Manager on the specified project. For example: **MyProjects/Project1/default.gpj**
- **Build Project** — Builds the specified project. For example: **MyProjects/Project1/default.gpj**
- **Debug Program** — Opens the MULTI Debugger on the specified executable. For example: **MyProjects/Project1/prog**
- **Debugger Command** — Performs the specified Debugger command. For example: **cb**

For a list of commands, see the *MULTI: Debugging Command Reference* book.

- **Editor** — Opens the MULTI Editor on the specified file. For example: **MyProjects/Project1/readme.txt**
- **Diff View** — Opens the Diff Viewer on two files or two versions of a file. For example: **MyProjects/Project1/default.gpj -v 1.94**
MyProjects/Project1/default.gpj -v 1.93

Argument usage for the **Diff View** action appears in a dialog box when you choose **Diff View** (Windows) or when your mouse hovers over the **Files** text field (UNIX).

- **Checkout Browser** — Opens the **Checkout Browser** on the specified directory. For example: **MyProjects/Project1**

- **Connection** — Connects to the specified target. For example: **simplpc**
- **Serial Terminal** — Connects via a **Serial Terminal** to the specified target. For example: **com1** (Windows) and **ttyS0** (Linux)
- **EventAnalyzer** — Opens the **EventAnalyzer** on the specified event source file. For example: **evlog.mes**
- **ResourceAnalyzer** — Connects the **ResourceAnalyzer** to the specified target. For example: **128.128.128.0**
- **Flash GUI** — Opens the **Write to Flash Memory** window. You may specify a program for the window to open on. For example: **C:\My Projects\Project1\catalog**

This action requires that MULTI be connected to a target.

- **Python Statement** — Executes the specified Python command(s). For example: **print "MULTI dir: \$_multi_dir";**

(\$_multi_dir is a Launcher variable. For more information, see “Using Variables in the MULTI Launcher” on page 115.)

- **Python Script** — Runs the specified Python script file. For example, **checkout_src.py**
- **Nonblocking Command** — Runs the specified shell command. The command output is not displayed. A nonblocking command is particularly good for launching graphical programs. For example: **gmemfile program**
- **Blocking Command** — Runs the specified shell command. Subsequent actions must wait until the **Blocking Command** has completed before running. The command output is displayed in an editor window. A blocking command is particularly good for command line programs or batch-processing applications. For example: **gsize program**
- **Wait** — Causes the Launcher to wait the specified number of milliseconds before executing the next action. For example: **10000** (waits for 10 seconds)
- **Utility Actions** — Runs one of the Green Hills utility programs. When you choose this action in the **Edit Action** dialog, the Launcher displays a list of utility actions and allows you to choose one. For example: **gstrip program**

If the chosen action type requires a filename or directory to be specified, the drop-down list will also contain a **Select File** or **Select Directory** option to launch a file or directory chooser.

In the argument string of an action, you can use MULTI Launcher variables to make the action general. For more information, see “Using Variables in the MULTI Launcher” on page 115.

Running Actions and Action Sequences

To run action sequences or actions, do one of the following:

1. Select one or more objects (either action sequences *or* actions) in the Launcher’s action tree, and select **Edit** → **Run Selected Objects**.
2. Right-click the selected object(s) (either action sequences *or* actions) in the Launcher’s action tree, and select **Run** from the shortcut menu.

To see example action sequences and actions, refer to “Overview of Workspaces and Shortcuts” on page 100.

If the actions or action sequences being executed contain a **Build Project** action, a **Python Statement** action, a **Python Script** action, a **Blocking Command** action, or any **Utility Action**, a progress window opens by default. For more information about the progress window, see “Managing Running Actions” on page 112.

Managing Running Actions

Each time one of the following workspace actions starts running, the corresponding process appears in the Launcher **Utilities** → **Running Actions** submenu:

- **Build Project**
- **Python Statement**
- **Python Script**
- **Blocking Command**
- Any **Utility Action**

If enabled [default], a MULTI Editor progress window displays when any of the preceding actions are run. The Launcher’s toggle menu item **Config** → **Show Progress Window** enables and disables the appearance of the progress window for these actions.

The title of a progress window indicates the currently executing action or, if execution is complete, the last executed action. Inside the progress window, action output is displayed in the normal text color and progress information is displayed in a special color.

If the execution of an action is not yet finished when you try to close the progress window, a dialog box asks if you want to terminate the execution.

You can also terminate execution of any action listed in the **Utilities** → **Running Actions** menu by selecting the corresponding menu entry. This menu does not list components of the MULTI IDE that were launched by Launcher workspace actions.



Note The **Utilities** → **Running Actions** menu only lists workspace actions. This menu is not related to programs being debugged by the MULTI Debugger.

MULTI Shortcuts

This section contains detailed information about creating, saving, and loading MULTI shortcuts. For general information about shortcuts, see “Overview of Workspaces and Shortcuts” on page 100.

Creating MULTI Shortcuts

MULTI shortcuts are like action sequences, except that they are not associated with any particular workspace.

To create a shortcut:

1. Click the **Shortcut** button ()
2. Select **Create or Change Shortcut**.

If it's not showing already, the **MULTI Launcher Shortcuts** pane appears. The **Working Dir** field is empty to indicate that shortcuts do not have a working directory associated with them. (Shortcuts execute in the working directory of the currently selected workspace.)

3. You can create or modify shortcuts just as you would action sequences for a workspace. See “Creating or Modifying an Action Sequence” on page 107.

Saving Shortcuts

To save your shortcuts to the default location, select **File → Save Changes** from the Launcher. The Launcher saves definitions of shortcuts into the shortcut configuration file and into **index.gmb**, which is located at:

- Windows — *user_dir*\Application Data\GHS\launcher
- UNIX — *user_dir*/.ghs/launcher

To find the full path to **index.gmb**, make a change to a shortcut and select **File → Save Changes** from the Launcher. The dialog box that opens displays the full path.

To save your shortcuts to a different file, see “Importing and Exporting Shortcuts” on page 114.

Importing and Exporting Shortcuts

If you want to export your shortcuts, select **File → Save Shortcuts into File** from the Launcher. In the file chooser that opens, select the file in which you want to save your shortcuts. You can share this file with other users.

To import shortcuts, select **File → Load Shortcuts from File** and choose a **.gmb** file. MULTI prompts you to choose whether you want to:

- Append these shortcuts to the list that appears when you select ,
- Replace the shortcuts currently listed when you select , or
- Cancel the load operation.

Using Variables in the MULTI Launcher

You can use variables to make actions and shortcuts generic and to make your projects more portable. This section begins with an overview of variable features and is followed by information about variable types; creating, modifying, and referencing variables; and variable substitution.

Overview of Variables

Launcher variables enable you to easily perform actions from the Launcher. For example, with the pre-defined `_ws_file_path` variable (described in “Variable Types” on page 116), you can create a shortcut that opens the Editor on the current workspace’s configuration file.

The values of generic Launcher variables, such as the pre-defined MULTI Launcher variables, change depending on the context of the workspaces where they are used. This property allows you to use a single variable in multiple workspaces. For more information about pre-defined MULTI Launcher variables, see “Variable Types” on page 116.

Generic variables also make your project more portable. If you associate a workspace with a particular project file and share the project with another user, you can share the workspace too. Simply use the `$_ws_file_dir` variable described in “Variable Types” on page 116 to specify the workspace’s working directory. When the second user opens the project and workspace, the value of the working directory dynamically updates to the correct location.

Variables are convenient to use. Typing a variable is usually much easier and faster than typing a long command string or pathname. In addition to being easy to type, variables allow you to maintain multiple workspaces simultaneously: modify a variable in one location, and you will see the change take effect every place where it is used.

You can use Launcher variables in the following places:

- Shortcuts and/or workspace actions — You can use variables when creating the workspace actions used in action sequences or shortcuts. For more information, see “Creating or Modifying an Action” on page 109.
- Workspace working directories — To use a variable to specify the working directory, click the **Properties** button in the Launcher. (If this button is not visible, click  to reveal the detail pane.) In the **Set Workspace Properties** dialog box, click the **Current** tab and enter the variable in the **Working Dir** field.
- Variable values — You can create variables for the current workspace or for all workspaces. For information about how to do this, see “Creating or Modifying Variables” on page 117.

Variable Types

The variables that the *MULTI* Launcher supports are categorized into the following types:

- *Workspace (or local) variables* — Available in the enclosing workspace. For information about creating workspace variables, see “Creating or Modifying Variables” on page 117.
- *Global variables* — Available to all workspaces. For information about creating global variables, see “Creating or Modifying Variables” on page 117.
- *Pre-defined *MULTI* Launcher variables* — Available to all workspaces and automatically defined by the *MULTI* Launcher. These variables and their values are listed on the **Others** tab of the **Set Workspace Properties** dialog box. For more information, see “The Detail Pane” on page 97.

The complete list of pre-defined variables follows.

- `_cwd` — Specifies the current working directory of the *MULTI* Launcher.
- `_ws_name` — Specifies the name of the current workspace.
- `_ws_working_dir` — Specifies the working directory of the current workspace.
- `_ws_file_name` — Specifies the base name of the file in which the current workspace is stored.

- `_ws_file_path` — Specifies the full path to the file in which the current workspace is stored.
 - `_ws_file_dir` — Specifies the directory of the file in which the current workspace is stored.
 - `_multi_dir` — Specifies the directory where the MULTI Launcher executable resides. This is in the MULTI installation directory.
 - `_user_cfg_dir` — Specifies the directory that stores the user's personal MULTI configuration.
 - `_multi_major_version` — Specifies the major version number of MULTI. For example, 5 would be the `_multi_major_version` in MULTI 5.0.1.
 - `_multi_minor_version` — Specifies the minor version number of MULTI. For example, 0 would be the `_multi_minor_version` in MULTI 5.0.1.
 - `_multi_micro_version` — Specifies the micro version number of MULTI. For example, 1 would be the `_multi_micro_version` in MULTI 5.0.1.
- Host environment variables — Available to all workspaces and defined in the host environment.

Creating or Modifying Variables

To create or modify a workspace variable or global variable:

1. Click the **Properties** button in the Launcher. If this button is not visible, click  to reveal the detail pane.
2. In the **Set Workspace Properties** dialog box, click the **Current** tab to create or modify the current workspace's variables and the **Global** tab to create or modify global variables.
3. To add a variable, click **New**. To edit a variable, select the variable and click **Edit**.
4. In the **MULTI Launcher Variable** dialog box, enter or modify the variable's name and its value.
5. Click **OK** in the **MULTI Launcher Variable** dialog box and in the **Set Workspace Properties** dialog box.

To delete a variable, select it in the **Set Workspace Properties** dialog box, and click **Delete**.

Workspace variables are a part of the workspace's definition and are kept in the workspace's definition file. Global variables are not a part of any workspace's definition and are kept in the **index.gmb** file located at:

- Windows — *user_dir\GHS\Launcher\index.gmb*
- UNIX — *user_dir/.ghs/Launcher/index.gmb*

Referencing Variables

To reference a variable, enter the dollar sign (\$) followed by the variable's name. If the variable name contains characters other than alphabetic characters, numeric characters, and underscore, enclose the variable name in parentheses () or square brackets []. Example variables follow:

- `$MyVar`
- `$(My Var)`
- `[$My Var]`

To precede a non-variable with a dollar sign (\$), escape the dollar sign with a backslash character (\). For example: `\$NotVar`.

Variable Substitution

When the MULTI Launcher evaluates a string containing variables, it scans the string from beginning to end and substitutes the string's variables with their values. The substitution process continues until no variables are left to substitute or until the Launcher has performed 20 scans. This maximum scan limit helps to prevent infinite loops caused by cyclic dependencies.

When the MULTI Launcher recognizes a variable name during the substitution process, it looks for the variable's value by searching the variable types in this order:

1. Workspace variables
2. Global variables
3. Pre-defined MULTI Launcher variables

4. Host environment variables

Because of the order in which the Launcher searches variable types, you can override a pre-defined or host environment variable by defining a workspace variable or global variable with the same name. To return back to the original value, delete the variable you created.

Using Version Control with MULTI

Chapter 6

This Chapter Contains:

- Configuring Version Control in MULTI
- Using Version Control with the Editor and Project Manager
- Integrating with Third-Party Version Control Systems
- Troubleshooting

You can integrate the MULTI IDE with any of a number of different version control systems: ClearCase, CVS, PVCS, RCS, SourceSafe, or Subversion.

The first portion of this chapter describes general procedures for enabling, configuring, and using a version control system with MULTI. The later sections provide specific instructions regarding each of the supported version control systems.



Note MULTI Version Control (MVC) has been removed from MULTI 5. Please contact Green Hills Technical Support if you need assistance transitioning from MVC to a new version control system.

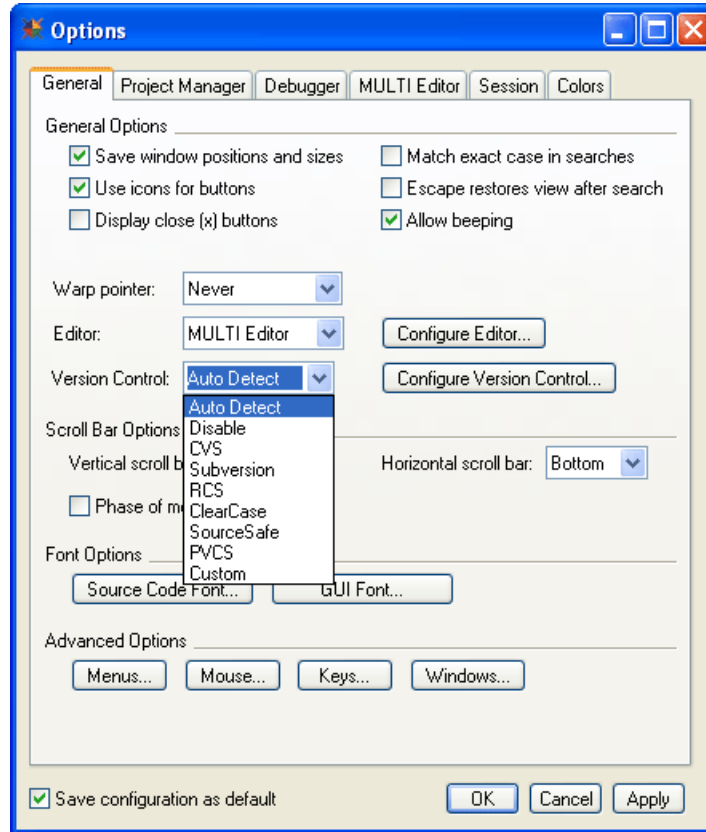
Configuring Version Control in MULTI

MULTI supports the following version control systems:

- ClearCase
- CVS
- PVCS
- RCS
- SourceSafe
- Subversion

MULTI's support for version control is enabled by default and will attempt to automatically detect which version control system is being used. To configure MULTI's version control support, perform the following steps:

1. Open the **Options** window:
 - In the Editor, Checkout Browser, Debugger or Launcher, select **Config** → **Options**; or
 - In the Project Manager, select **Tools** → **Options**.
2. Select the **General** tab of the **Options** window.
3. In the **Version Control** drop-down list, select a version control system.



The available options are:

- **Auto Detect** — Causes MULTI to automatically detect the version control system being used. This is the default setting, and should only be changed if MULTI does not integrate correctly with your version control system.
- **Disable** — Disables MULTI's version control integration.
- **CVS** — Configures MULTI to integrate with CVS (Concurrent Version Systems). For detailed information, see “Integrating with CVS” on page 129.
- **Subversion** — Configures MULTI to integrate with Subversion. For detailed information, see “Integrating with Subversion” on page 134.
- **RCS** — Configures MULTI to integrate with the RCS version control system. For detailed information, see “Integrating with RCS” on page 131.

- **ClearCase** — Configures MULTI to integrate with the Rational ClearCase version control system. For detailed information, see “Integrating with ClearCase” on page 128.
 - **SourceSafe** (Windows only) — Configures MULTI to integrate with the Microsoft SourceSafe version control system. For detailed information, see “Integrating with SourceSafe” on page 132.
 - **PVCS** (Windows only) — Configures MULTI to integrate with the MERANT PVCS version control system. For detailed information, see “Integrating with PVCS” on page 130.
4. Click the **Configure Version Control** button to open a dialog box that will prompt you to enter settings that are specific to the version control system you have selected. After you have entered the applicable settings, click **OK** in the dialog box.
 5. Click **OK** in the **Options** window. For information about saving your changes, see “Saving Configuration Settings” on page 172.



Note You may need to restart any Editors or Project Managers that you have open before the version control configuration changes take effect.

Using Version Control with the Editor and Project Manager

The MULTI Editor and MULTI Project Manager contain menus that provide access to common version control actions, such as checking files in and out and viewing version history.

- In the Editor, version control actions are available from the **Version** menu.
- In the Project Manager, version control actions are available from the **Tools** → **Version Control** submenu.

The options on these menus vary slightly. Both menus are described below.

Version Control Options from the Editor

The following menu items are available in the MULTI Editor’s **Version** menu. Their availability is determined by the version control system used. Check the section corresponding to the version control system you are using to determine

which items are available and any special notes on the item specific to your version control system.

Check Out	Check a file out of the version control system to allow for editing. If the file is not checked out, no edits will be allowed. This is not meaningful for CVS or Subversion.
Update	Issue an update command. If the repository version corresponding to the version of the local file has changed, the changes are merged into the local file. This is only available for CVS and Subversion.
Check In/Commit	Check a file into a version control system. Commit is used for CVS and Subversion, and Check In is used for other version control systems.
Check In All	Check in all checked out files at once.
Discard Changes	Discard current changes and release the file if it was checked out.
Place Under VC	Place a file under a version control system. If a version control system was not specified, it will be automatically detected based on the version control system used for other files in the same directory.
Auto Checkout	When an edit operation is desired and this option is checked, the file will automatically be checked out. This is not meaningful for CVS or Subversion.
Show History	Starts the History Browser on the file, see “The History Browser” on page 154. In ClearCase, this starts the ClearCase history browser.
Show Last Edit	Shows the most recent edit of text in the selected region. Some version control systems cannot show deletions as the last edit. See “Show Last Edit” on page 127.
Revert to History	Starts the History Browser. The file is reverted to a specific version chosen through the History Browser. See “The History Browser” on page 154.
Revert to Date	Prompts for a date and reverts the file to that date.
Revert to Version	Prompts for a version and reverts the file to that version.
Show View	Shows both the working directory and the set ClearCase view. The working directory used is the directory of the current file being edited. This menu item is only available for ClearCase on non-Windows hosts.

Version Control Options from the Project Manager

These are the items that can be accessed through the **Tools** → **Version Control** menu in the Project Manager.

Check Out	Check the selected files out of a version control system to allow for edit. If the file is not checked out, no edits will be allowed. This is not meaningful for CVS or Subversion.
Check In/Commit	Check the selected files into a version control system. Commit is used for CVS and Subversion, and Check In is used for other systems.
Check In + Out	Perform the Check In operation and then the Check Out operation.
Retrieve	Retrieves read-only copies of the selected files.
Discard Changes	Discard current changes for the files selected and release any selected files that are checked out.
Place Under VC	Place the selected files under a version control system. If a version control system was not specified, it will be automatically detected.
Show History	Starts the History Browser on the selected files. In ClearCase, this starts the ClearCase history browser. See “The History Browser” on page 154.
Checkout Browser	Starts the Checkout Browser . See “The Checkout Browser” on page 138.
Other VC Command	Opens a dialog which allows a version control command to be called.

Automatic Checkout

If you enable the **Automatic Checkout** option, the MULTI Editor will automatically check out a file when you start editing it. In most version control systems, files that have not been checked out are read-only, which prevents you from making changes to them. CVS and Subversion are exceptions to this rule because they allow multiple users to edit the same file simultaneously and then they attempt to merge the changes.

To configure the Editor to automatically check out files when you modify them:

1. In the Editor, select the **Config** → **Options**. In the Project Manager, select **Tools** → **Options**.

2. Select the **General** tab.
3. Click the **Configure Version Control** button and check the **Automatic Checkout** option if available (see “Version Control Configuration Options” on page 215). This box may not be available for certain version control systems. For example, in CVS a file is always checked out so there is no option to enable or disable **Automatic Checkout**.



Note To control **Automatic Checkout** on a per-file basis, select **Version → Auto Checkout** in the Editor.

Show Last Edit

To show the most recent edit made to a portion of a file:

1. In the Editor, select the portion of the file you want to examine.
2. Select **Version → Show Last Edit**.

This finds the last version of the current file in which the selected text changed (or the entire file, if no text is selected) and displays the difference in the Diff Viewer. For information about the Diff Viewer, see “The Diff Viewer” on page 157. If the selection has never changed, a dialog stating this will be displayed.

This command is only available if the file is under a version control system (such as CVS, ClearCase, SourceSafe, and Subversion) that supports this feature. This command is also available if the file has been locally modified, in which case the Diff Viewer will display the difference between the local file and the repository version of the file.



Note When using CVS, ClearCase, SourceSafe, or Subversion, **Show Last Edit** does not detect deletions. Because lines that were deleted in past versions cannot be selected, the **Show Last Edit** menu item does not show deletions as the most recent change.

Revert to a Previous Version of a File

There are three ways to specify the version of a file to which you want to revert:

- To select the version to revert to from a list of all versions of the file:
 1. Select **Version** → **Revert to History**.
 2. A History Browser will open. You can right-click a version in the browser window and select **Revert to Version** (see “The History Browser” on page 154).
- To revert to a version checked in on a specific date:
 1. Select **Version** → **Revert to Date**.
 2. A dialog box will open where you can enter the date of the appropriate version.
- To revert to a specific version number:
 1. Select **Version** → **Revert to Version**.
 2. A dialog box will open where you can enter the appropriate version number.

Integrating with Third-Party Version Control Systems

The following sections describe how to integrate with third-party version control systems supported by MULTI.

Integrating with ClearCase

If you use ClearCase, the program **cleartool** should exist in your path. If it does not, or if your ClearCase program uses a filename other than **cleartool** you should configure your ClearCase options to set the path to the ClearCase executable (see “Configuring Version Control in MULTI” on page 122). On Windows, the default location is **C:\Program Files\Rational\ClearCase\bin\cleartool**. On UNIX, the default location is **/usr/atria/bin/cleartool**.

The following features and issues are specific to ClearCase:

- **Show History** starts the ClearCase history browser.
- **Revert To History** and **Revert To Date** are not available.

- The **Show View** item is added to the Editor **Version** menu. (See “Working with Views” on page 129).
- The **Show Last Edit** displays the last modification to the selected lines. Because lines that were deleted in past versions are not selected, the **Show Last Edit** menu item does not show deletions as the most recent change.
- There is no **Checkout Browser** support for ClearCase.

Working with Views

If you are going to set a view, it must be set before MULTI is started. There is no way to set or change the view once MULTI has been started. To see your current view, choose **Version** → **Show View** from the Editor. This will display the working directory view and the set view. The working directory view is also displayed next to the filename being edited at the top of the editor pane.

If it is necessary to work with two different views then set a view, start the Editor, and then set the next view and start another Editor.

Integrating with CVS

If you use CVS (Concurrent Version Systems), the program **cv**s should exist in your path. If it does not, or if your CVS program uses a filename other than **cv**s, configure your CVS options to set the path to the CVS executable (see “Configuring Version Control in MULTI” on page 122).

The following features and issues are specific to CVS:

- **Show Last Edit** displays the last modification to the selected lines. Because lines that were deleted in past versions are not selected, the **Show Last Edit** menu item does not show deletions as the most recent change.
- The **Checkout Browser** is supported with CVS. For more information, see “The Checkout Browser” on page 138.
- The History Browser is supported with CVS. For more information, see “The History Browser” on page 154.
- Version control integration may not function optimally in a Cygwin environment. For more information, see “Using Subversion or CVS in a Cygwin Environment (Windows)” on page 135.

Integrating with PVCS



Note The PVCS integration is only available on Windows hosts with PVCS version 8.

The following features and issues are specific to PVCS:

- **Show Last Edit** is not available.
- The **Checkout Browser** is supported with PVCS. For more information, see “The Checkout Browser” on page 138.
- The History Browser is supported with PVCS. For more information, see “The History Browser” on page 154.

Assumptions

The following assumptions are made.

It is assumed that the `workfile` location for a project database is set at the root level and that all the subprojects are assigned default `workfile` locations. For example, if the `workfile` location for a project database is `C:\project1`, then the subproject `subproject1` is assigned `C:\project1\subproject1` as its `workfile` location.

It is assumed that there is a local copy of the project—including subproject—database files located at the project database `workfile` location. The integration with the **Checkout Browser** relies on comparing the directory structure of the project database `workfile` location and the project database.

It is assumed the login source for the project database is either **VCSID** or **HOME**.

Most of the version control operations (such as **Check Out**, **Place Under VC**) assume read/write access permissions. These permissions are assumed both for the `workfile` location and for the archive directory. However, you can view files read only.

Required Setup Steps

It is necessary to specify the project database `workfile` location, otherwise known as the root of the checkout. It is also necessary to specify what project database is going to be used. There are two ways to specify these parameters.

1. The first way to configure these parameters is through the GUI.
 - a. Select the **Config** → **Options** available in most MULTI GUIs.
 - b. Select the **General** tab and then choose **PVCS** from the **Version Control** pull-down.
 - c. Click the **Configure Version Control** button and enter the path to the root of the checkout and the full path of the master configuration file for the project database being used. Ask your administrator for the location of this file.
2. The second way to configure these parameters is to create a **.pvcsdir** file at the root of each project database checkout. In other words, this file has to be located in the project database `workfile` location. This file should contain a single line with the full pathname of the master configuration file for the project database being used. Ask your administrator for the location of this file. The location of the file specifies the root of the checkout and the content of the file specifies what project database is going to be used.

It is necessary to use this method if the auto detection option is enabled. Auto detection works by locating and reading the **.pvcsdir** file.

Using the Integration

To create a new subproject in the project database, simply create a new directory in the appropriate place in the `workfile` location and start checking in files. With the first checked in file, a new subproject will be created by that name.

Integrating with RCS

If you use RCS, the programs **ci**, **co**, and **rlog** must exist in your path.

With RCS, a file is read-only until it is checked out. Usually a file is checked out (**Check Out**), modified, and checked back in (**Check In**). If the **Auto Checkout** option is enabled, the file is automatically checked out when an edit is attempted (see “Automatic Checkout” on page 126).

The following features and issues are specific to RCS:

- **Show Last Edit** is not available with RCS.
- The **Checkout Browser** is not supported with RCS.

Integrating with SourceSafe



Note The SourceSafe integration is only available on Windows hosts.

If you use SourceSafe, the program **ss.exe** should exist in your path. If it does not, or if your SourceSafe program uses a filename other than **ss.exe** you should configure your SourceSafe options to set the path to the SourceSafe executable (see “Configuring Version Control in MULTI” on page 122). The default location used is **C:\Program Files\Microsoft Visual Studio\VSS\win32\ss.exe**.

One advantage of this integration is that several different databases can be used simultaneously. The user can edit files from two different databases at the same time. The integration requires some setup steps to indicate what database is being used.

The following features and issues are specific to SourceSafe:

- **Show Last Edit** displays the last modification to the selected lines. Because lines that were deleted in past versions are not selected, the **Show Last Edit** menu item does not show deletions as the most recent change.
- The **Checkout Browser** is supported with SourceSafe. For more information, see “The Checkout Browser” on page 138.
- The History Browser is supported with SourceSafe. For more information, see “The History Browser” on page 154.

VSS Database Structure Assumptions

The following assumptions are made about the structure of the Visual Source Safe (VSS) database.

A working directory is defined. This should be defined using the **Visual SourceSafe Explorer**, which comes with the installation of VSS.

The directory structure in the working directory mirrors the directory structure of the database. For example: If there is a database with the following structure:

```
$/  
|_hello  
|_hello.c
```

and if the working directory of `$/` is `C:\ssafedir`, then the working directory of the hello project is assumed to be `C:\ssafedir\hello`.

Most of the version control operations (such as **Check Out**, **Place Under VC**) assume read/write access permissions. However, you can view files read only.

Required Setup Steps

1. It is necessary to specify the database working folder location, otherwise known as the root of the checkout. It is also necessary to specify what database is going to be used. There are two ways to specify these parameters.
 - a. The first way to configure these parameters is through the GUI.
 - i. Select the **Config** → **Options** available in most MULTI GUIs.
 - ii. Select the **General** tab and then choose **SourceSafe** from the **Version Control** pull-down.
 - iii. Click the **Configure Version Control** button and enter the path to the root of the checkout and the directory path of the **srcsafe.ini** file for the database being used. Ask your administrator for the location of this file.
 - b. The second way to configure these parameters is to create a **.ssdir** file at the root of each database checkout. This file should contain a single line with the directory path to the **srcsafe.ini** file for that particular database. For the example in the previous section it would be necessary to place a **.ssdir** file in the directory `C:\ssafedir`. The contents of the file would look similar to:

```
C:\Program Files\Microsoft Visual Studio\VSS
```

Ask your administrator for the location or determine the location by using the **Open SourceSafe Database** option in the **Visual SourceSafe Explorer**. This is a one time setup step required for each database checkout you want to use the version control integration with.

It is necessary to use this method if the auto detection option is enabled. Auto detection works by locating and reading the **.ssdir** file.

2. Set the **SSUSER** environment variable. Set **SSUSER** to your SourceSafe username. This is necessary to avoid queries for your username.
3. Set the **SSPWD** environment variable. Set **SSPWD** to the SourceSafe password that corresponds to the **SSUSER** environment variable. This is necessary to avoid queries for your password.

Using the Integration

To create a new project in the database, simply create a new directory in the appropriate place in the working directory and start checking in files. With the first checked in file, a new project will be created by that name.

Integrating with Subversion

If you use Subversion, the program **svn** should exist in your path. If it does not, or if your Subversion program uses a filename other than **svn**, configure your Subversion options to set the path to the Subversion executable. For information about how to do this, see “Configuring Version Control in MULTI” on page 122.

The following features and issues are specific to Subversion:

- **Show Last Edit** displays the last modification to the selected lines. Because lines that were deleted in past versions are not selected, the **Show Last Edit** menu item does not show deletions as the most recent change.
- The **Checkout Browser** is supported with Subversion. For more information, see “The Checkout Browser” on page 138.
- The History Browser is supported with Subversion. In addition to the normal History Browser functions, the comment area located at the bottom of the window displays commit information related to the selected file. Namely, if another file or files were committed at the same time as the selected file, the History Browser displays these additional filenames. This information can be useful if you want to see what other files contain related changes.

The History Browser precedes each additional filename with a letter indicating the action that was taken before the file was committed. Possible letters and their meanings follow.

- **A** — The file was added to the repository before being committed.
- **D** — The file was deleted from the repository before being committed.
- **M** — The file already existed in the repository and was modified before being committed.

For more information, see “The History Browser” on page 154.

- Version control integration may not function optimally in a Cygwin environment. For more information, see “Using Subversion or CVS in a Cygwin Environment (Windows)” on page 135.

Troubleshooting

Using Subversion or CVS in a Cygwin Environment (Windows)

If you use Cygwin as your primary shell, you may encounter problems when trying to perform Subversion or CVS version control operations from the MULTI tools. By default, Cygwin installs its own versions of **svn.exe** and **cvs.exe**, which are heavily modified to work within the Cygwin environment. Additionally, Cygwin modifies your PATH environment variable so that a search for **cvs.exe** or **svn.exe** first finds Cygwin’s copies. By default, MULTI is configured to search your PATH and use the first **cvs** or **svn** executable it finds. To force use of the correct executable, we recommend that you configure MULTI to reference the full path of the non-Cygwin **cvs.exe** or **svn.exe**. To do so, follow the steps listed in “Configuring Version Control in MULTI” on page 122. You can enter the full path to the executable in the **CVS Configuration** or **Subversion Configuration** dialog box that opens when you perform step 4.

Version Control Tools

Chapter 7

This Chapter Contains:

- The Checkout Browser
- The Commit Changes Dialog Box
- The History Browser
- The Diff Viewer

MULTI provides three graphical tools for viewing information about files under version control:

- **Checkout Browser** — Provides a graphical view of the files in your checkout and easy access to version control operations, such as checking files in and out and viewing revision history. For more information, see “The Checkout Browser” on page 138.
- **History Browser** — Displays the complete revision history of any file under version control. For more information, see “The History Browser” on page 154.
- **Diff Viewer** — Compares two ASCII files (typically different revisions of the same file) and highlights differences between them. For more information, see “The Diff Viewer” on page 157.

Each of these tools is described in detail below.

The Checkout Browser

The **Checkout Browser** is designed to be used with CVS, PVCS, SourceSafe, or Subversion to give you a graphical view of your checkout and the status of files in the repository. While the **History Browser** (see “The History Browser” on page 154) is designed to give you detailed version information about a specific file, the **Checkout Browser** gives you general information about every file in your checkout, and provides access to more detailed information about any file within the checkout. The **Checkout Browser** also allows you to manipulate the files in your checkout. For example, you can select files in the **Checkout Browser** that have been modified, compare those files using the **Diff Viewer**, then commit those files to the repository.



Note The term *repository* is used throughout this chapter to mean the central store of version control information for the version control system in use. Different version control systems may use different terms, for example, in SourceSafe the repository is called a *database*.

Starting the Checkout Browser

You can start the **Checkout Browser** any of the following ways:

- From the **Launcher**, click the **Checkout Browser** button () or select **Components** → **Open Checkout Browser**. (For more information about the **Launcher**, see “The MULTI Launcher” on page 4).
- On the command line, enter **mcobrowse.exe** (Windows) or **mcobrowse** (UNIX).
- From the **History Browser**, select **File** → **Checkout Browser**.

Using the Checkout Browser

The basic modes of operation for the **Checkout Browser** are:

- Scanning a preexisting checkout
- Modifying an existing checkout (CVS only)
- Creating a new checkout (CVS only)

This section explains how to scan a preexisting checkout. For information about modifying an existing checkout or creating a new checkout, see “Using the CVS Checkout Editor” on page 149.

Scanning loads the information about an existing checkout or part of a checkout into the Checkout Browser. To scan a preexisting checkout:

1. Set the **Roots** field. The **Roots** field indicates what directories will be scanned and displayed in the browser. You can use any of the following methods to select what will be displayed:
 - Enter a directory in **Roots**. When using CVS or Subversion, it is possible to supply a comma-separated list of directories.
 - Use the **Roots** drop-down list to select from directories that have previously been displayed in the **Checkout Browser**.
 - Click  to browse for the correct folder.

2. Once you have selected the directories to be displayed, use the **File** menu to select one of the following two types of scans:

- **Local Rescan** — Provides local information, such as which files have been locally modified. Local scans do not contact the remote repository, and as a result are usually faster than full scans. This also means that they do not tax the remote repository, which may have many users. This type of scan is only available for CVS and Subversion.

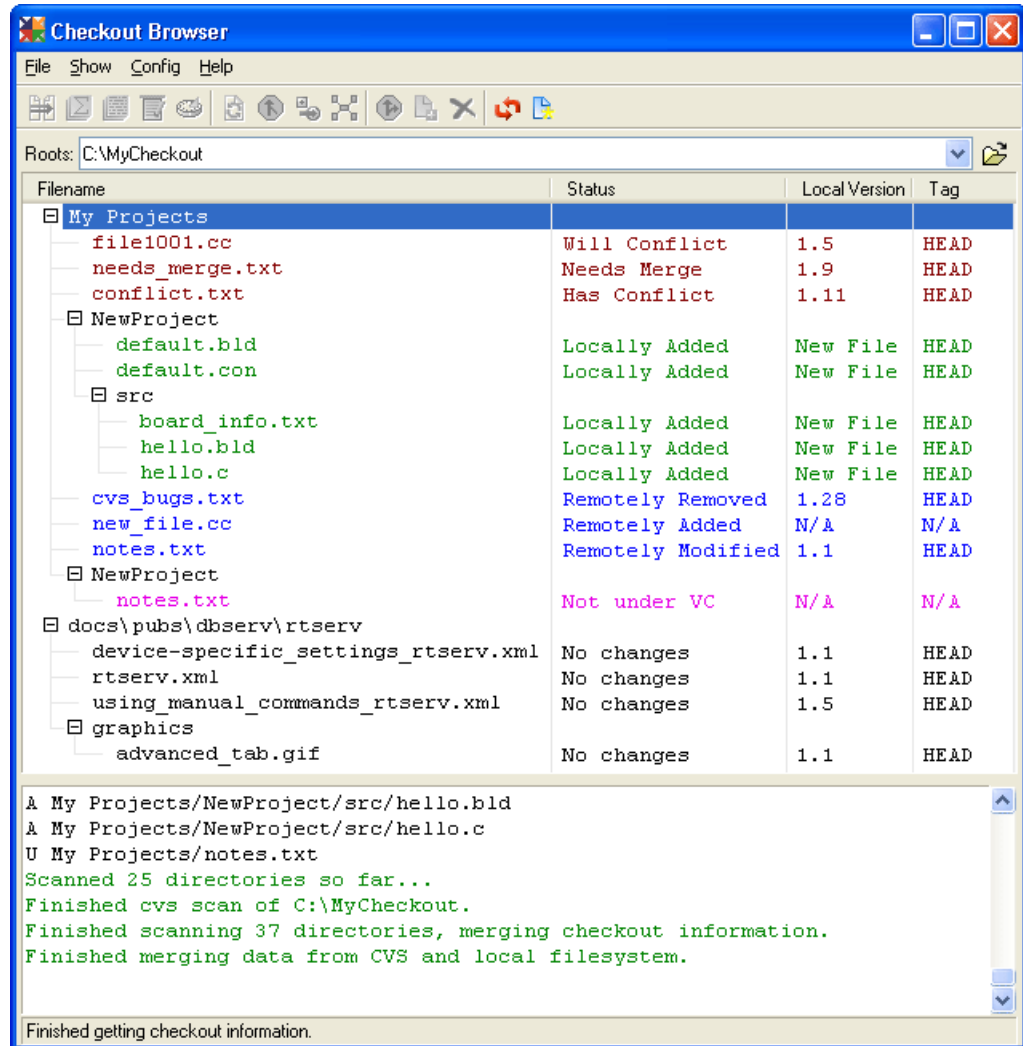


Note **Local Rescan** does not support **.cvsignore** files. If you want information about these files, select **File** → **Full Rescan**.

- **Full Rescan** — Provides information about local and remote changes to files. Because full scans gather information from the remote repository, they usually take longer than local scans.

Pressing Enter after entering or selecting a directory in the **Roots** field will also perform a full scan.

3. The **Checkout Browser** will scan the selected directories and display the results of the scan.



The following information is displayed in the Checkout Browser.

Filename

Names of files in the folder you selected. By default, the Checkout Browser only displays changed files. For information about changing the default, see “The Show Menu” on page 146.

You can click the  and  buttons to expand or contract the file list.

Status

Current version control status of the file. The possible values for the status are:

- **No changes** — The file has not been changed.
- **Locally Modified** — Changes have been made to the file in your checkout that do not exist in the remote repository. There is a chance that updating the file will result in conflicts if changes have been made to the repository since your last scan.
- **Locally Added** — The file exists in your checkout and has been marked to be added to the remote repository when committed.
- **Locally Removed** — The file has been marked to be removed from the remote repository when committed. The file no longer exists in your checkout.
- **Remotely Modified** — The version in the repository is newer than the version in your checkout, perhaps because another user has checked in a new version of the file.
- **Remotely Added** — The file exists in the remote repository, but is not in your checkout. This is most likely because someone else has added the file to the repository.
- **Remotely Removed** — The file has been removed from the repository, but is still in your checkout.
- **Not under VC** — The file is not recognized by the version control system. This identifies files that have been created inside your checkout, but have not been added or marked to be added to the repository.
- **Needs Merge** — The file has been modified in your checkout, but a newer version exists in the repository than the version you started with. You must update your checkout to merge the changes between your local version and the repository version before you can commit your changes to the repository. There is a chance that the process of updating the file will result in a conflict.
- **Will Conflict** — Similar to **Needs Merge**, except when you update your checkout, the merge will create conflicts that must be resolved before you will be able to commit the file to the remote repository.
- **Has Conflict** — The file in your checkout contains conflicts that must be resolved before you will be able to commit it to the remote repository.
- **Generated by Conflict** — The file was generated by the version control system when it updated a different file, which resulted in a conflict. When the conflict is resolved, this file is removed. Only available for Subversion.

Local Version
Version of the file as it currently exists in the local checkout.
Tag
The branch tag, if applicable.
Locked By
A list of user names that have the file locked. Only available for SourceSafe and PVCS.

The Checkout Browser Toolbar

The **Checkout Browser** toolbar contains buttons that allow you to perform various version control operations on selected files. The following table describes the buttons in the **Checkout Browser**.

Button	Effect
 Diff with local version	Opens a Diff Viewer that shows differences between the locally modified version of the selected files and the versions in the repository that match the values in the Local Version field. If the files have not been changed in the repository since the last time the files were updated, the Local Version is also the latest version in the repository. If the files have since been changed, Local Version indicates particular file versions in the repository that the locally modified versions are compared against. To compare against the latest versions in the repository in this case, use Diff with remote version, described below.
 Diff with remote version	Opens a Diff Viewer that shows differences between the latest version in the repository and your local version. This button is only available for files that have been remotely modified (files with Status listed as Remotely Modified, Needs Merge, or Will Conflict).
 Diff all locally modified files	Opens a Diff Viewer showing differences between local and original versions of all modified files.
 Edit	Opens the Editor on the selected file(s).

Button	Effect
 Show history	Opens a History Browser that displays information about all versions of the selected file (see “The History Browser” on page 154). If you select multiple files, a separate History Browser opens for each file.
 Rescan	Updates the status of the selected files or directory. This is useful for determining whether either the repository or local versions have been modified since the last scan.
 Update	Brings the selected files or directory in your checkout up to date with the versions in the repository.
 Commit	Checks in local changes of the files or directories selected, creating new versions in the repository. This launches the Commit Changes dialog box. For more information, see “The Commit Changes Dialog Box” on page 152.
 Commit all locally modified, added, and removed files	Checks in changes for all the files or directories that have been locally modified, added, or removed, creating new versions in the repository. This launches the Commit Changes dialog box. For more information, see “The Commit Changes Dialog Box” on page 152.
 Revert	Reverts the selected file to its previous state. For example, if you select a locally added file and then click this button, the file is removed from the repository and returns to the Not Under VC state. If you select a locally modified file and then click this button, local changes are removed and the file reverts to the version located in the repository.
 Add	Marks files to be added to the version control system. Files will not be added to the remote repository until you Commit them.
 Delete	Marks files to be deleted from the version control system. If the files selected are not under version control, they will be deleted from your checkout. If the files exist in the repository, they will be removed from your checkout and marked for deletion. They will not be removed from the repository until you Commit them.

Button	Effect
 Rescan All	Updates the status of all files and directories located in the Roots directory. This is useful for determining whether either the repository or local versions have been modified since the last full scan.
 Add Checkout	Allows you to add another directory to the current Checkout Browser . A window in which you can specify the additional directory appears when you click the button.

The Checkout Browser Menus

The following menus and menu items are available from the **Checkout Browser**.

The File Menu

The following table describes the options available from the **File** menu.

New Window
Opens another Checkout Browser .
Launcher
Opens the MULTI Launcher .
Full Rescan
Gathers information from the remote repository and provides information about modified files. After this scan the Checkout Browser shows the status of each file in your checkout, and it shows the corresponding repository.
Local Rescan
Provides local information, such as which files have been locally modified. After this scan the Checkout Browser will show as much status information for each file in your checkout as it can determine without contacting the repository. Only available for CVS and Subversion.
Note: Local Rescan does not support .cvsignore files. If you want information about these files, select File → Full Rescan .
Update Checkout
Brings the local checkout up to date with the remote repository, performs a scan for local changes, and displays the results in the Checkout Browser .

Halt Scan / Update Stops any Rescan or Update in progress.
CVS Checkout Editor Opens the CVS Checkout Editor . This feature is only supported when using CVS. You can use the CVS Checkout Editor to create a new checkout or to modify the contents of an entire checkout or portion of a checkout, such as reverting a checkout to a certain date. For more information, see “Using the CVS Checkout Editor” on page 149.
Close Closes the Checkout Browser .

The Show Menu

The following table describes the options available from the **Show** menu. This menu is used to select what type of files are displayed in the **Checkout Browser**. A check mark appears next to items that are currently selected for display. Selecting a menu item toggles the display on or off for that particular type of file. You can choose to display any combination of file types from the menu.

No Changes Displays files with no changes.
Conflicts Displays files with conflicts that must be resolved before you can commit your local version to the repository (files with Status listed as Needs Merge , Will Conflict , or Has Conflict).
Local Changes Displays files that have been changed locally.
Remote Changes Displays files that have been changed remotely.
Not Under VC Displays files that are not currently under Version Control.

The Config Menu

The **Checkout Browser**’s **Config** menu contains the same menu items that appear in the **Config** menus of other MULTI tools. You can use this menu to

modify configuration options for MULTI. For a description of each **Config** menu item, see “The Config Menu” on page 73.

The Checkout Browser Shortcut Menu

When you right-click files or directories in the **Checkout Browser**, a shortcut menu appears. The menu displays some or all of the following options, depending on which version control system you use and what files or directories you have selected. For example, if you select a directory, the **Show History** and **Diff** menu options will be unavailable. Similarly, you cannot **Commit** a file if no changes have been made, **Add** a file that already exists in the repository, or **Update** a file that does not contain any remote changes.

Diff with Local Version

Opens a **Diff Viewer** that shows differences between the local files and the versions of the files listed in the **Local Version** column. See “The Diff Viewer” on page 157.

Diff with Remote Version

Opens a **Diff Viewer** that shows differences between the latest version in the repository and your local version. This menu item is only available for files that have been remotely modified (files with **Status** listed as **Remotely Modified**, **Needs Merge**, or **Will Conflict**).

Edit

Opens an editor on the selected files.

Show History

Opens a **History Browser** that displays information about all versions of the file (see “The History Browser” on page 154). If you select multiple files, a separate **History Browser** will open for each file.

Rescan

Updates the status of the selected files or directory. Useful for determining whether either the repository or local versions have been modified since the last full scan.

Update

Brings the selected files or directory in your checkout up to date with the versions in the repository.

Commit / Check In

Checks in local changes of the files or directory selected, creating new versions in the repository. This launches the **Commit Changes** dialog box. For more information, see “The Commit Changes Dialog Box” on page 152.

Check Out
Checks out the selected files from the repository locking them. Only available for SourceSafe and PVCS.
Resolve Conflict
Clears the conflict state of the selected files. As a side effect, this may remove files that were generated by the conflict. Only available for Subversion.
Unlock
Releases the lock on the selected files. Only available for SourceSafe and PVCS.
Place Under VC
Adds the selected files to the repository, creates an initial version, and checks that version out. Only available for SourceSafe and PVCS.
Revert to Repository
Reverts the selected file to its previous state. For example, if you select a locally added file and then select this option, the file is removed from the repository and returns to the Not Under VC state. If you select a locally modified file and then select this option, local changes are removed and the file reverts to the version located in the repository.
Add
Marks files to be added to the version control system. Files will not be added to the remote repository until you Commit them.
Delete
Marks files to be deleted from the version control system. If the files selected are not under version control they will be deleted from your checkout. If the files exist in the repository they will be removed from your checkout and marked for deletion. They will not be removed from the repository until you Commit them.

The Status Bar and Status Window

The **Checkout Browser** performs many operations that can take a long time. Information about the operation the **Checkout Browser** is performing at any given time is displayed in two places, both located in the bottom portion of the window.

- The status pane displays a continuously updated stream of status messages when the **Checkout Browser** performs modify, scan, or create operations. For CVS and Subversion, these messages include the number of directories that have been scanned and the current output from the CVS or Subversion

executable. Different information may be displayed for other version control systems.

- The status bar lists current activity when the **Checkout Browser** is performing operations such as committing files.

Features and Issues Specific to CVS

There are some limitations in the **Checkout Browser** when you use it with CVS. These limitations are:

- Updating the entire checkout (or a subdirectory) automatically uses the **-Pd** option to pull in new directories from the repository and prune empty directories from the checkout.
- Directories that have been added to the repository do not display after a **Rescan** although they will be displayed after an **Update**. This is because the **Checkout Browser** cannot determine whether the new directories are empty, and will be pruned with the next update.

Using the CVS Checkout Editor

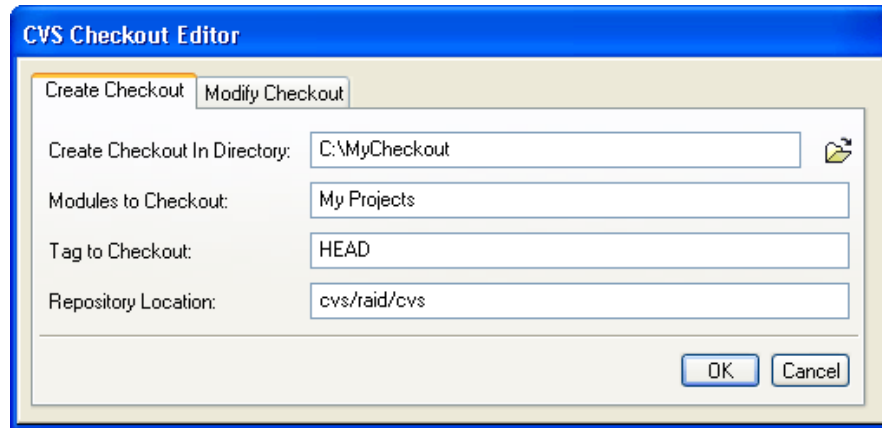
The **Checkout Browser** provides the capability to create or modify an entire CVS checkout with the **CVS Checkout Editor**.

To open the **CVS Checkout Editor**, select **File → CVS Checkout Editor** from the **Checkout Browser**.

The **CVS Checkout Editor** has two modes of operation, one for creating a new checkout and one for modifying an existing checkout.

To create a new checkout:

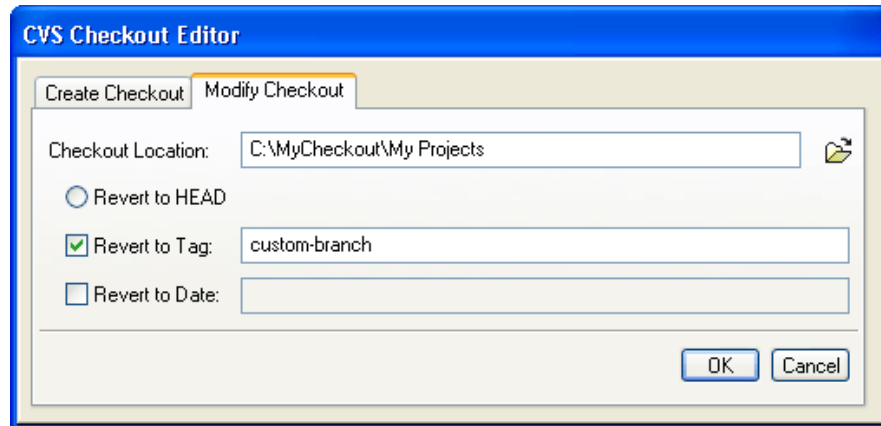
1. Select the **Create Checkout** tab.



2. Fill in the required fields:
 - **Create Checkout In Directory** — Enter the destination directory, which is where the checkout will be created. If this directory does not exist, but its parent does, the directory will be created automatically.
 - **Modules to Checkout** — Enter a space-separated list that specifies what should be checked out. You can enter filenames, directories, and/or modules as defined on the CVS server.
 - **Tag to Checkout** — Enter the tag to use when creating the checkout.
 - **Repository Location** — Enter the location of the repository. By default, the repository location is set to whatever is in the `$CVSROOT` environment variable.
3. Click **OK** to create the checkout.

To modify an existing checkout:

1. Select the **Modify Checkout** tab.



2. Enter the directory of the checkout to be modified in the **Checkout Location** text field. This may be a subdirectory of the root of the checkout.
3. Select either **Revert to HEAD** or one or both of **Revert to Tag** and **Revert to Date**:
 - **Revert to HEAD** — Select this option to reset all sticky tags. (Selecting this radio button clears the **Revert to Tag** and **Revert to Date** check boxes.)
 - **Revert to Tag** — Select the check box, then enter the tag of the checkout to revert to in the text field. (Selecting this check box clears the **Revert to HEAD** radio button.)
 - **Revert to Date** — Select the check box, then enter the date of the checkout to revert to in the text field. CVS supports many different date formats. Example date formats are 2006-09-24 and 24 Sep 2006. (Selecting this check box clears the **Revert to HEAD** radio button.)

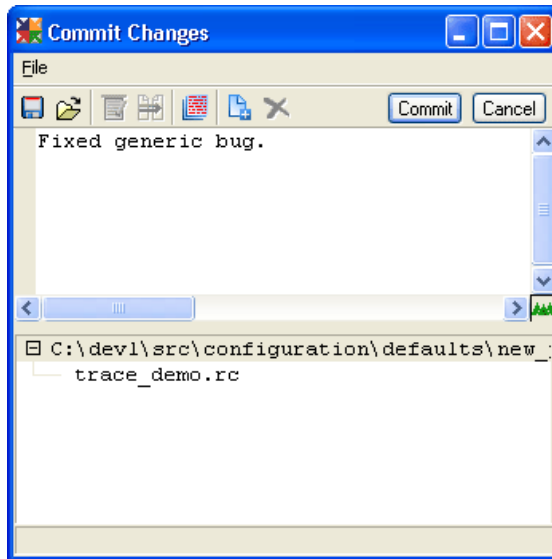


Note If you specify both a tag and a date, CVS sets the sticky tag. It also sets the version back to the specified date in that tagged branch, but does not set the sticky date.

4. Click **OK** to modify the checkout.

The Commit Changes Dialog Box

The **Commit Changes** dialog box allows you to specify a log message when committing one or more files. The dialog box also gives you the ability to manipulate which files are committed and to see the changes in those files. A sample **Commit Changes** dialog box follows.



The **Commit Changes** dialog box opens when you try to commit one or more files via the **Checkout Browser** or the MULTI Editor. To commit files, do one of the following:

- In the **Checkout Browser**, right-click a locally modified file or a selection of locally modified files, and select **Commit**.
- In the MULTI Editor, select **Version** → **Check In** or **Version** → **Commit** (the option varies depending on your version control system).

When the **Commit Changes** dialog box initially opens, the bottom pane contains a list of files to commit. Depending on the specifics of your version control system and its configuration, the text field at the top of the dialog box may contain a template for a commit message. Simply enter a commit message and click the **Commit** button to commit the selected files. If the commit operation fails, the dialog box is not dismissed, giving you the opportunity to address whatever issue caused the failure without losing your commit message. The following sections describe the various menu entries and buttons in the dialog box.

Commit Changes Options

The following menu items are available from the **File** menu in the **Commit Changes** dialog box. Some of these menu items are also present in the shortcut menu that appears when you right-click a file or directory. Where applicable, corresponding toolbar buttons are displayed beside menu items.

Save Commit Log 
Opens a dialog box that you can use to save the current log message to a file.
Import Commit Log 
Opens a dialog box that allows you to import a log message. The imported message replaces the currently displayed log message (if any).
Use Recent Commit Log
Opens a submenu that displays shortened versions of recent commit log messages. Selecting one of these messages replaces the current message (if any).
Add New Files 
Opens a dialog box from which you can choose additional files to be committed.
Remove Selected Files
Removes the selected file(s) from the list of files to commit. If no files are selected in the bottom portion of the dialog box, this option is unavailable.
Edit Selected Files 
Opens the Editor on the selected file(s). If no files are selected in the bottom portion of the dialog box, this option is unavailable.
Diff Selected Files 
Opens a Diff Viewer on the selected files. The Diff Viewer shows differences between the local file and the version of the file that was originally checked out. See “The Diff Viewer” on page 157.
If no files are selected in the bottom portion of the dialog box, this option is unavailable.
Close 
Closes the Commit Changes dialog box.

The History Browser

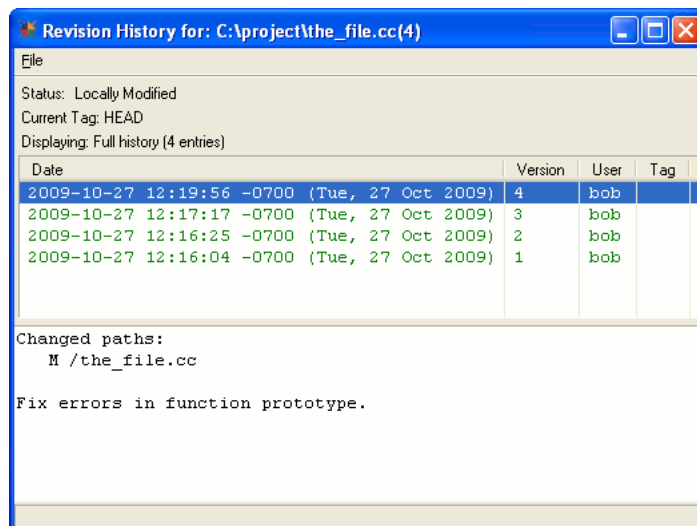
The **History Browser** allows you to examine all versions of a single file that exist in your version control system. As you scroll through the versions of the file, you can view the check in comments associated with those versions. You can also use the **History Browser** to compare versions or revert to a previous version.

Using the History Browser

To start the **History Browser**, do one of the following:

- In the Editor, select **Version** → **Show History**.
- In the Project Manager, select **Tools** → **Version Control** → **Show History**.
- In the **Checkout Browser**, click the **Show History** button (🍰).
- You can open the **History Browser** as a stand-alone application by starting the MULTI Editor with the **-historybrowser filename** option. For example, to start a **History Browser** that lists the version history of the source files **sample.cc**, enter:

```
me -historybrowser sample.cc
```



The **History Browser** displays the following information:

Status Current version control status of the file.
Current Tag Identifying label that was last used to update your local machine. A version might have several labels associated with it in the version control system, but only the label used to update the file on your local machine will appear.
Displaying Shows the number of history entries and whether a Partial history or a Full history is being displayed. If you are using Subversion and there are more than 100 history entries, Partial history is shown. In this case, the History Browser also displays the Get more and Get all buttons, which retrieve up to 100 additional entries or all entries, respectively.
Version List The top half of the History Browser window displays a table containing the following information for all versions logged in version control: • Date — Lists the date and time each version was checked into the repository. • Version — Lists the revision number assigned to each version of the file. If possible, version numbers are grouped by branches with the +/- construct. • User — Lists the user who saved each version. • Tag — Lists the tags corresponding to each version. This is only available for CVS. Information is displayed sorted by Version. You can change the sort criteria by clicking any of the column headings.
Comment area The bottom half of the History Browser window displays comments associated with the version selected in the top half of the window. It also lists tags associated with this version.



Note To view version changes that occurred after the **History Browser** was opened, select **File** → **Refresh History**.

The History Browser File Menu

The **History Browser** has one menu, the **File** menu, which contains the following items:

Diff with Local Version
Opens a Diff Viewer that shows the differences between the selected version of the file and your local version of the file. See “The Diff Viewer” on page 157.
Diff 2 Versions
Opens a Diff Viewer that shows the differences between two selected versions of the file. If you select a range of versions, Diff 2 Versions will perform a diff between the upper and lower bounds of the range. See “The Diff Viewer” on page 157.
View Selected Version
Opens the selected version of the file in an editor.
Revert to Selected Version
Overwrites the local copy of the file with the selected version.
Edit Local Version
Opens the local copy of the file in an editor.
Refresh History
Updates the information in the History Browser .
Checkout Browser
Opens the Checkout Browser . See “The Checkout Browser” on page 138.
Close
Closes the History Browser .

The History Browser Shortcut Menu

The following shortcut menu can be accessed by right-clicking a version in the **History Browser**:

Diff with Local Version
Opens a Diff Viewer that shows differences between the selected version of the file and your local version of the file. See “The Diff Viewer” on page 157.
Diff 2 Versions
Opens a Diff Viewer that shows difference between the two selected versions of the file. If you select a range of versions, Diff 2 Versions will performs a diff on the upper and lower bounds of range. See “The Diff Viewer” on page 157.

View

Opens the selected version of the file in an editor.

Revert to

Overwrites the local copy of the file with the selected version.

The Diff Viewer

The **Diff Viewer** graphically displays differences between two ASCII files, whether they are different files or different versions of the same file.

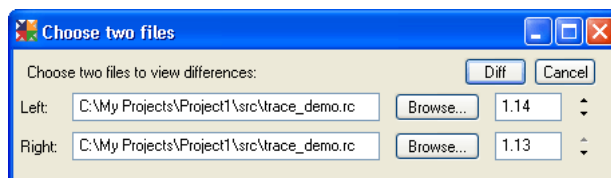
The following bullet points describe the ways in which you can open the **Diff Viewer**:

- MULTI automatically opens the **Diff Viewer** whenever you choose to compare two files or two file versions in the **History Browser** or **Checkout Browser** (see “The History Browser” on page 154 and “The Checkout Browser” on page 138).
- The **Diff Viewer** is a stand-alone tool that you can open from the command line with one of the following executables:
 - Windows — **diffview.exe**
 - UNIX — **diffview**

For required and optional command line options, see “Opening the Diff Viewer from the Command Line” on page 158.

- In the Debugger command pane, enter the **diff** command.
- In the Editor, select **Tools** → **Diff Files** (see “The Tools Menu” on page 68).

If you have not already specified the pair of files or file versions to compare, the **Choose two files** dialog box opens:



Select filenames and versions to compare, then click **Diff**. The **Diff Viewer** opens on the specified files or file versions. For more information, see “Using the Diff Viewer” on page 160.

Opening the Diff Viewer from the Command Line

You can use the **diffview** command to launch the **Diff Viewer** from the command line. The **diffview** executable is located in the Green Hills installation directory.

The **Diff Viewer** accepts the following command line arguments, which control the files and versions compared.

- The CVS-style syntax is:
diffview [*options*] [-r *rev1* | -D *date1*] [-r *rev2* | -D *date2*] *file1*
- The Subversion-style syntax is:
diffview [*options*] [-r *rev1*[:*rev2*]] *file1* [*file2*]...

The following table describes the **Diff Viewer** options.



Note In this table, the term *whitespace* refers to space and tab characters. It does not refer to newline characters.

-add
Adds the files or file versions to the pane in the lower-left corner of the Diff Viewer , but only displays them if there are no other diffs.
-b
Ignores differences in the amounts of whitespace when displaying differences. See also -lang=c and -w .
-c
Shows only three lines of context around differences.
-h, --help (UNIX only)
Displays information about diffview command options.
-i
Ignores differences in case.
-ignore_numbers
Ignores differences in decimal numbers.

-intraline

Displays differences within lines.

See also **-nointraline**.

-intraline_lineup

Displays differences within lines column by column, not allowing insertions or deletions.

See also **-nointraline_lineup**.

-intraline_tokens

Displays differences within lines word by word (default).

See also **-nointraline_tokens**.

-lang=c

Ignores whitespace using heuristic C tokenization.

See also **-b** and **-w**.

-local

Diff's a list of files against their local versions.

-new, -noreuse

Opens a new Diff Viewer window.

See also **-reuse**.

-nointraline

Does not display differences within lines.

See also **intraline**.

-nointraline_lineup

Displays differences within lines, allowing insertions or deletions (default).

See also **intraline_lineup**.

-nointraline_tokens

Displays differences within lines character by character.

See also **intraline_tokens**.

-reuse

Reuses an existing **Diff Viewer** window (default).

See also **-new**, **-noreuse**.

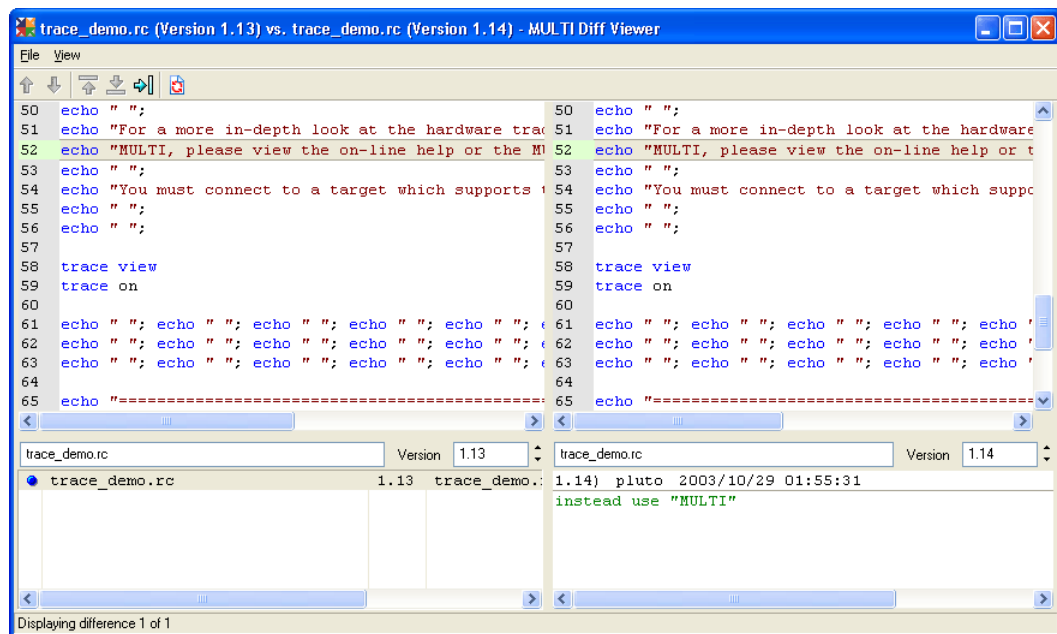
-w

Ignores all whitespace.

See also **-b** and **-lang=c**.

Using the Diff Viewer

The **Diff Viewer** contains the following features:



- The main panes of the **Diff Viewer** display the pair of files or file versions that are being compared. Differences are highlighted in the control area color, while the current selection is highlighted in the highlight color.
- The status bar displays the current diff number and total number of diffs.
- The pane in the lower-right corner displays the version control comments associated with the file displayed in the right-hand pane. This pane is not present if you are displaying local changes or if the file is not under version control.

- The pane in the lower-left corner contains a list of all the file or file-version pairs currently available for diffing in the **Diff Viewer**. The columns in this pane display the following information (from left to right):
 - The first column displays a blue dot next to the current file or file-version pair. The **Reload** () button can be used to reload the current diff from the files on disk.
 - The second column shows the file that is displayed in the left pane when the row is selected, and the third column shows the version of that file. If the file is on disk, this field displays **disk**; if the file has been locally modified, it displays **local (version number)** to indicate the version of the file that it is based on.
 - The fourth column shows the file that is displayed in the right pane when the row is selected, and the fifth column shows the version of that file. If the file is on disk, this field displays **disk**; if the file has been locally modified, it displays **local (version number)**.
 - The last column displays the total number of differences. A question mark (?) in this column indicates that the number of differences has not been calculated. To improve performance, the number of differences is calculated only when the diff for that row is displayed.

Right-click in the lower-left pane to access a shortcut menu that contains the following operations:

- **Show Diff** — Displays the highlighted diff view in the main panes of the **Diff Viewer**. The blue dot appears next to the file/file-version pair you selected.
- **Remove Diff** — Removes the highlighted diff view from the pane in the lower-left corner, and from the main panes of the **Diff Viewer**, if it is currently displayed.

Navigating the Diff Viewer

The **Diff Viewer** provides several ways to navigate through files:

- The  button and the minus (-) keyboard key jump up to the next difference between the two displayed files/file versions.
- The  button and the plus (+) keyboard key jump down to the next difference.
- The  button jumps to the first difference.
- The  button jumps to the last difference.
- The  button and the asterisk (*) keyboard key reposition the pane to display the current difference (this is useful if you lose your place while scrolling).
- The Editor keyboard shortcuts Ctrl + F and Ctrl + B incrementally search forward or backward through the files/file versions (see “Incremental Searching” on page 28). If there is no current selection, the **Diff Viewer** begins its search at the top left corner of the visible pane.
- The Editor keyboard shortcut Ctrl + G goes to a line number (see “Go To a Line Quickly” on page 26).

Diff Viewer Menus

The following **Diff Viewer** actions can be executed via the **File** menu or the keyboard shortcuts listed.

Open Diff (Ctrl + O)
Opens a diff file selection dialog box that adds a new diff to the current Diff Viewer window.
Open New Diff (Ctrl + Shift + O)
Opens a new diff file selection dialog box that shows the new diff in a new Diff Viewer window.
Show Current Diff (Ctrl + *)
Re-positions the Diff Viewer panes to the current diff (highlighted in the highlight color).
Close (Ctrl + Q)
Closes the Diff Viewer .

You can use the **Diff Viewer** in several different modes to alter what types of differences it shows. The following modes can be set via the **Diff Viewer's View** menu.



Note In the following table, the term *whitespace* refers to space and tab characters. It does not refer to newline characters.

Ignore Whitespace Amount

Ignores differences in the amounts of whitespace between two otherwise identical lines. For example, in this mode, the following text is considered to be the same:

```
void func(int a);
```

and

```
void      func(int      a);
```

If one line uses ten spaces while its counterpart uses only one, this difference is not considered significant. However, if the second line uses no whitespace, a difference is shown.

You can only select one whitespace option at a time.

Ignore Whitespace for C

Ignores whitespace using heuristic C tokenization of a file. Whitespace differences within a line that do not affect compilation do not show up as differences. For example, in this mode, the following text is considered to be the same:

```
void func(int a);
```

and

```
void func      ( int      a ) ;
```

Because it does not have enough information available to perform a true compilation of the file, the heuristic parser that C tokenization uses cannot perfectly account for all cases.

You can only select one whitespace option at a time.

Ignore All Whitespace

Ignores all whitespace differences between files. For example, in this mode, the following text is considered to be the same:

```
voidfunc(inta);
```

and

```
void func (int a);
```

You can only select one whitespace option at a time.

Case Insensitive

Ignores all differences in capitalization between lines. For example, in this mode, the following text is considered to be the same:

```
void func(int a);
```

and

```
void Func(int A);
```

This mode can be selected along with one of the whitespace modes.

Display Changes Within Lines

Highlights differences within non-matching lines, making it easier to identify changes in each line. Portions of the line that match appear in dark gray.

If this option is not selected, the entire line is highlighted if it is different than the corresponding line in the other pane.

Display Changes Word by Word Within Lines

Divides each line into words before displaying changes within lines. This option can make changes within lines easier to read, even when two different words happen to share one or more letters.

This option has no effect unless **Display Changes Within Lines** is also selected.

Line up Columns for Changes Within Lines

Treats characters as if they are different if they appear in different columns. This option can be useful when you compare files in which column position is significant (S-Record files, for example).

This option has no effect unless **Display Changes Within Lines** is also selected.

Ignore Changes in Numbers (0–9)

Ignores all differences in decimal numbers between lines. For example, in this mode, the following text is considered to be the same:

```
stw    r4, 8(r9)
```

and

```
stw    r5, 4(r8)
```

This mode can be selected along with one of the whitespace modes.

The Diff Viewer Shortcut Menu

When you right-click in either of the two main **Diff Viewer** panes, a shortcut menu provides the following options:

Edit File on Disk

Opens the selected file in a text editor.

Show Last Edit

Diff's the right-clicked version of the file against the most recent version of the file in which the selected text changed. Both panes update as needed to display applicable versions.

This option only applies to files that are under version control. It only appears when you have selected a region of text.

Show Last Edit in File

Diff's the right-clicked version of the file against the most recent version of the file that changed. Both panes update as needed to display applicable versions.

This option only applies to files that are under version control. It only appears when you have *not* selected a region of text.

Show History

Opens a **History Browser** for the selected file. See “The History Browser” on page 154.

This option only applies to files that are under version control.

Set to Current Diff

Specifies that the diff containing the cursor is the current diff (highlighted in the highlight color).

Configuring the MULTI IDE

Part III

Configuring and Customizing MULTI

Chapter 8

This Chapter Contains:

- Setting Configuration Options
- Creating Custom Functionality Using Scripts and Macros
- Customizing the GUI
- Configuring and Customizing Toolbar Buttons
- Configuring and Customizing Menus
- Customizing Keys and Mouse Behavior
- Configuring Taskbar Organization
- Configuring Window Docking
- Configuring Window Focus
- Configuring File Extensions
- Configuring File Associations (Windows only)

This chapter explains how you can configure and customize the MULTI environment to suit your specific needs and preferences. It includes information about:

- Setting configuration options that affect how various components of the MULTI IDE function (see “Setting Configuration Options” on page 170).
- Using scripts and/or macros to automate tasks and create custom functionality (see “Creating Custom Functionality Using Scripts and Macros” on page 179).
- Customizing the appearance and behavior of MULTI’s graphical components (see “Customizing the GUI” on page 181).

Setting Configuration Options

MULTI contains a number of configuration options that control how various components of the MULTI IDE function. The following sections explain how you can change the settings of these options via the **Options** window, the **configure** command, and configuration files.

Using the Options Window

To set or edit configuration options via the **Options** window, do one of the following:

- From the Launcher, Debugger, or Editor, select **Config → Options**.
- From the Project Manager, select **Tools → Options**.

The **Options** window opens, allowing you to edit the values of most of the configuration options that affect MULTI. For information about each configuration option and its available settings, see Chapter 9, “Configuration Options”.

By default, any changes you make in the **Options** window are remembered across MULTI sessions. If you want changes to affect only the current MULTI session, clear the check box labeled **Save configuration as user default** (located in the bottom-left corner of the **Options** window).

Using the **configure** Command

You can set or edit configuration options from the MULTI Debugger command pane by issuing the **configure** command. Configuration settings entered or modified with the **configure** command are automatically reflected in the **Options** window.

The **configure** command may take any of the following forms:

- **configure** *config_option=value*
- **configure** *config_option: value*
- **configure** *config_option value*

where *config_option* identifies which option you are setting and *value* is an appropriate setting for that option. For example:

```
> configure tabsize=9
> configure moon: On
> configure linenumbermode: Both
> configure prompt: "Your wish is my command> "
> configure background #ffffff
```

If *value* is a Boolean, you may also enter:

- **configure** *config_option*

to toggle the Boolean value. MULTI indicates the new value. For example:

```
> configure closebuttonontitlebar
Toggling closebuttonontitlebar: new value is Off
```

To display a list of all the *config_options* that can be set, enter:

- **configure ?**

For a complete description of every configuration option and the available settings for each, see Chapter 9, “Configuration Options”.

By default, changes you make to configuration settings via the **configure** command only affect the current MULTI session. For information about saving configuration modifications, see the next section.

Saving Configuration Settings

Changes you make to configuration settings via the **Options** window or the **configure** command are saved to configuration files (*.cfg). You can save changes to the user configuration file, the global configuration file, an application-specific configuration override file, or a file you specify and that you can manually load in future sessions. The following sections describe how to save configuration settings to each of these files.

Saving to the User Configuration File

To save configuration changes to the user configuration file so that they are automatically loaded every time you start a MULTI application, do one of the following:

- From most MULTI applications, select **Config** → **Save Configuration as User Default**. (From the Project Manager, select **Tools** → **Configuration** → **Save Configuration as User Default**.)
- Enter the **saveconfig** command (see “General Configuration Commands” in Chapter 7, “Configuration Command Reference” in the *MULTI: Debugging Command Reference* book).

Configuration settings modified from the **Options** window are saved to the user configuration file by default. For more information, see “Using the Options Window” on page 170.

For more information about the user configuration file, see “The User Configuration File” on page 176.

Saving to the Global Configuration File

To save your current settings to the global configuration file so that they are loaded for every user of the installation:

1. Enter the **saveconfigtofile** command in the Debugger command pane, or select **Config** → **Save Configuration As** from most MULTI applications (from the Project Manager, select **Tools** → **Configuration** → **Save Configuration As**). For more information about the **saveconfigtofile** command, see “General Configuration Commands” in Chapter 7,

“Configuration Command Reference” in the *MULTI: Debugging Command Reference* book.

2. The **Save Configuration to what file?** dialog box opens. Save the file as:
 - Windows — *install_dir\config\ghs.cfg*
 - UNIX — *install_dir/config/ghs.cfg*

For more information about the global configuration file, see “The Global Configuration File” on page 175.

Saving to an Application-Specific Configuration Override File

To save your current settings to an application-specific override file so that the global configuration file is overridden for a particular MULTI application:

1. Enter the **saveconfigtofile** command in the Debugger command pane, or select **Config** → **Save Configuration As** from most MULTI applications (from the Project Manager, select **Tools** → **Configuration** → **Save Configuration As**). For more information about the **saveconfigtofile** command, see “General Configuration Commands” in Chapter 7, “Configuration Command Reference” in the *MULTI: Debugging Command Reference* book.
2. The **Save Configuration to what file?** dialog box opens. Save the file as:
 - Windows — *install_dir\config\override\application_name.cfg*
 - UNIX — *install_dir/config/override/application_name.cfg*

where *application_name* is the name of the application whose configuration you want to override. For example, to override the global configuration of the MULTI Editor, name the file **me.cfg**.

To save your current settings to an application-specific override file so that the user configuration file is overridden for a particular MULTI application, follow the preceding steps, but save the file as:

- Windows — *user_dir\Application Data\GHS\v5\override\application_name.cfg*
- UNIX — *user_dir/.ghs/v5/override/application_name.cfg*

Saving to a User-Specified Configuration File

To save your current settings to a configuration file that is not automatically loaded at startup but that you can manually load to configure new or existing MULTI sessions, perform the following steps:

1. Enter the **saveconfigtofile** command in the Debugger command pane, or select **Config** → **Save Configuration As** from most MULTI applications (from the Project Manager, select **Tools** → **Configuration** → **Save Configuration As**). For more information about the **saveconfigtofile** command, see “General Configuration Commands” in Chapter 7, “Configuration Command Reference” in the *MULTI: Debugging Command Reference* book.
2. In the **Save Configuration to what file?** dialog box, specify a path and filename for the configuration file.

See also “Loading Configuration Files” on page 178.

Propagating Configuration Settings

In general, if you change the setting of a configuration option but do not save the configuration as the default, only the MULTI application from which the option was changed is aware of the new setting. If you do save the configuration as the default, however, all MULTI applications launched after the change are also notified of the new setting.

A limited number of configuration options do not follow this general rule. Changing the setting of one of these options causes all MULTI applications opened during the current session to be notified of the change—whether or not you saved the configuration as the default. These options are noted where applicable.

To be sure that a new option setting takes effect in all MULTI applications, save the configuration as the default and close and restart the MULTI IDE.

Creating and Editing Configuration Files

As an alternative to setting configuration options via the **Options** window or the **configure** command, you can manually create or edit configuration files (*.cfg) using the MULTI Editor or another text editor. There are several types of configuration files:

- Global configuration file (see page 175)
- User configuration file (see page 176)
- Application-specific configuration override files (see page 176)
- Command line configuration file (see page 177)

If conflicting settings exist among these configuration files, the order in which the files are loaded determines which settings take effect. For more information, see “Loading Configuration Files” on page 178.

For information about the format of configuration files, see “Configuration File Format” on page 177.

The Global Configuration File

Changes you make to the global configuration file are read each time a MULTI application starts and affect every user of the installation. This file is useful for setting up a common environment for a group of users working with the MULTI IDE. The global configuration file should be located at:

- Windows — *install_dir\config\ghs.cfg*
- UNIX — *install_dir/config/ghs.cfg*

If **ghs.cfg** is not found, *application_name.cfg* is used (if it exists), where *application_name* is a replaceable for a MULTI stand-alone application. For example, **multi.cfg** is the .cfg file for the MULTI Debugger, **me.cfg** is the .cfg file for the MULTI Editor, etc.

The global configuration file can be overridden on a per-application basis. For more information, see “Application-Specific Configuration Override Files” on page 176.

The User Configuration File

Changes you make to the user configuration file are read each time a MULTI application starts and only affect the user specified in *user_dir* (see below). This file is useful for setting up the environment required by a single user working with the MULTI IDE. The user configuration file should be located at:

- Windows — *user_dir*\Application Data\GHS\v5\ghs.cfg
- UNIX — *user_dir*/.ghs/v5/ghs.cfg

If **ghs.cfg** is not found, *application_name.cfg* is used (if it exists), where *application_name* is a replaceable for a MULTI stand-alone application. For example, **multi.cfg** is the .cfg file for the MULTI Debugger, **me.cfg** is the .cfg file for the MULTI Editor, etc.

The user configuration file can be overridden on a per-application basis. For more information, see “Application-Specific Configuration Override Files” on page 176.

Application-Specific Configuration Override Files

Both the global configuration file and the user configuration file can be overridden on a per-application basis.

To override the global configuration file for a particular MULTI application, create a configuration file located in:

- Windows — *install_dir*\config\override
- UNIX — *install_dir*/config/override

The override configuration file should be named *application_name.cfg*, where *application_name* is the name of the application whose configuration you want to override. For example, to override the global configuration of the MULTI Editor, name the file **me.cfg**.

To override the user configuration file for a particular application, create a configuration file named *application_name.cfg* in:

- Windows — *user_dir*\Application Data\GHS\v5\override
- UNIX — *user_dir*/.ghs/v5/override

The Command Line Configuration File

You can specify a configuration file (*.**cfg**) from the command line with the **-configure** *filename* option (see “Using the Command Line” in Chapter 2, “Before You Begin Debugging” in the *MULTI: Debugging* book).

Configuration File Format

Each line in a configuration file should either:

- Begin with a pound sign (#) as the first non-whitespace character and contain a comment (these lines are ignored)
- Be in the format

config_option : *value*

where *config_option* identifies which option you are setting and *value* is an appropriate setting for that option. Appropriate values for *config_option* and *value* are the same as those used with the **configure** command (see “Using the configure Command” on page 171), but you do not need to include the **configure** command in **.cfg** files.

For a complete description of every configuration option and the available settings for each, see Chapter 9, “Configuration Options”.

Example 1. Configuration File Contents

```
# This is a comment that's ignored.
TabSize: 8
Background: #ffffff
Moon: On
LineNumberMode: Both
Prompt: "> "
# The following blank line is ignored as well:

AlwaysUseMeToFixBuildErrors: Off
```

Loading Configuration Files

On startup, MULTI applications automatically initialize configuration options from the following configuration files (*.cfg), if they exist, in the following order:

1. Global configuration file (If a global configuration override file exists for a particular application, the application loads it instead of the global configuration file. See page 175 and page 176.)
2. User configuration file (If a user configuration override file exists for a particular application, the application loads it instead of the user configuration file. See page 176 and page 176.)
3. The configuration file, if any, specified on the command line with the **-configure** option (See page 177.)

If conflicting settings exist among these configuration files, the settings in the last file loaded override settings specified in previously loaded files.



Note If you start the MULTI Editor from the MULTI Debugger, the Editor inherits the settings that were initialized when the Debugger was started. As a result, any configuration override files that exist to overwrite settings for the Editor will not take effect.

Script files can also affect configuration options. For more information, see “Using Script Files” on page 180.

To load a predefined configuration file during a MULTI session, do one of the following:

- Choose **Config** → **Load Configuration** (from the Project Manager, **Tools** → **Configuration** → **Load Configuration**), and select the configuration file you want to load.
- In the Debugger command pane, use the **configurefile** command (see “General Configuration Commands” in Chapter 7, “Configuration Command Reference” in the *MULTI: Debugging Command Reference* book).

Clearing Configuration Settings

To permanently delete your user configuration file, which contains configuration options you have saved, do one of the following:

- Select **Config → Clear User Default Configuration** from most MULTI applications (from the Project Manager, select **Tools → Configuration → Clear User Default Configuration**).
- Enter the **clearconfig** command in the Debugger command pane (see “General Configuration Commands” in Chapter 7, “Configuration Command Reference” in the *MULTI: Debugging Command Reference* book).
- Delete the following file:
 - Windows — `user_dir\Application Data\GHS\v5\ghs.cfg`
 - UNIX — `user_dir/.ghs/v5/ghs.cfg`

Future sessions will use MULTI’s default settings unless you create a new user configuration file or a user configuration override file (operates on a per-application basis) or specify another configuration file for the session. For more information about the user configuration file, see “The User Configuration File” on page 176.

Creating Custom Functionality Using Scripts and Macros

You can use scripts and/or macros to customize MULTI to simplify and automate tasks. A script is a list of commands and expressions in a file that MULTI reads and executes as if they were entered individually in the Debugger command pane.

Scripts are powerful tools for automating both common tasks and regression testing. A script file could, for example, compare a program variable to a constant value and then perform some action based on that result. Scripts are also useful for configuring your target board. You could also write a command script that executes parts of your program and then checks to see whether the process is running correctly. Such a script would be useful for verifying that the process still runs as expected after you make changes. For complete details on how to write, use, and debug scripts, see Chapter 2, “Using MULTI Scripts” in the *MULTI: Scripting* book.

Using Script Files

Upon startup, or upon loading a program, the MULTI Debugger can execute script files. Script files run after configuration files are loaded (see “Loading Configuration Files” on page 178). The Debugger runs the following scripts, if they exist, in the following order:

1. Global script file
2. User script file
3. The script file, if any, specified on the command line with the **-rc** option
4. Program script file

The global script file, user script file, and command line script file are executed when the first MULTI Debugger window appears. All Debugger windows launched in a debugging session share the same Debugger environment. The program script file is executed every time the corresponding program is loaded into a Debugger window.

The following table provides details about these script files.

Global script file • Windows — <i>install_dir</i>\config\multi.rc • UNIX — <i>install_dir</i>/config/multi.rc This file is useful for commands that need to run once when any person in a user group starts the Debugger for the first time in a debugging session.
User script file • Windows — <i>user_dir</i>\Application Data\GHS\multi.rc • UNIX — <i>user_dir</i>.ghs/v5/multi.rc This file is useful for commands that need to run once when a single user debugs any program.

Command line script file

You can specify a script file from the command line with the **-rc** option (see Appendix C, “Command Line Reference” in the *MULTI: Debugging* book). For example:

```
multi my_prog -rc my_script
```

Program script file

- Windows — **executable_dir\executable_name.rc**
- UNIX — **executable_direxecutable_name.rc**

The program script file is executed every time the corresponding program is loaded into a Debugger window (that is, with every new Debugger window opened on the program, or with every program reload in the current Debugger, but not with every process restart).

If you load a new program into an existing Debugger window, MULTI will automatically execute the commands in the script file of the new program (if any), but it does not clean up the effects of the script file of the old program (if any).

For more information about writing scripts to include in your startup files, see Chapter 2, “Using MULTI Scripts” in the *MULTI: Scripting* book.

Customizing the GUI

MULTI allows you to customize buttons, menus, keystroke combinations, and mouse clicks in the Debugger, Project Manager, Editor, and some other MULTI applications. You can remove these GUI elements, add new GUI elements, or change the commands that are executed when the GUI element is used.

You can define customized GUI elements using the following methods:

- Enter configuration commands in the Debugger command pane. For proper syntax, refer to the commands that appear in the following sections.
- Use the **Options** window to access and change configuration options. To open the **Options** window, select **Config** → **Options**.
- Write a script file that contains customization commands. The following table lists the commands used to configure each GUI element. Be aware that the script files that MULTI loads automatically (global, user, and program script files) are loaded only when the Debugger is launched, so do not put customizations that apply to the Project Manager and Editor into these startup script files; use a configuration file instead. For more information

about writing a script, see Chapter 2, “Using MULTI Scripts” in the *MULTI: Scripting* book.

Changes must be saved in a configuration file if you want the configuration available for later use (see “Saving Configuration Settings” on page 172). Once you have defined a configuration file or script file that customizes the GUI, you can have MULTI load that file. For more information about these and other files that MULTI uses at startup, see “Creating and Editing Configuration Files” on page 175 and “Using Script Files” on page 180.

To customize a particular GUI element, use the following as a guide:

Debugger buttons

- Debugger command pane — To customize buttons within the current MULTI session, use the **debugbutton** command. For information about the **debugbutton** command, see “Configuring and Customizing Toolbar Buttons” on page 183.
- GUI — To customize buttons across MULTI sessions, select **Config** → **Customize Toolbar** from the Debugger menu bar. For information about how to use the window that appears, see “Adding, Removing, and Rearranging Toolbar Buttons” in Appendix A, “Debugger GUI Reference” in the *MULTI: Debugging* book.

Editor buttons

- Debugger command pane — Use the **editbutton** command (see “Configuring and Customizing Toolbar Buttons” on page 183).
- GUI — Select **Config** → **Options** → **Editor** → **Configure Editor Buttons** (see “The MULTI Editor Options Tab” on page 251).

GUI menus

- Debugger command pane — Use the **menus** command (see “Configuring and Customizing Menus” on page 186).
- GUI — Select **Config** → **Options** → **General tab** → **Menus** (see “The General Options Tab” on page 208).

Keyboard shortcuts

- Debugger command pane — Use the **keybind** command (see **Customizing Keys and Mouse Behavior** on page 191).
- GUI — Select **Config** → **Options** → **General tab** → **Keys** (see “The General Options Tab” on page 208).

Mouse actions

- Debugger command pane — Use the **mouse** command (see **Customizing Keys and Mouse Behavior** on page 191).
- GUI — Select **Config** → **Options** → **General tab** → **Mouse** (see “The General Options Tab” on page 208).

Configuring and Customizing Toolbar Buttons

You can customize Debugger or Editor toolbar buttons by using the following Debugger commands:

debugbutton [*num*] [*name*] [*c=command*] [*i=iconname*] [*h=helpstring*] [*t=tooltip*]

editbutton [*num* | *name*] [*c=command*] [*i=iconname*] [*h=helpstring*] [*t=tooltip*]

Adds, deletes, or configures a button on the Debugger’s (**debugbutton**) or Editor’s (**editbutton**) toolbar, where:

- *num* is the number that MULTI assigns to the button.
- *name* is the name of the button.
- *command* is the command executed when the button is pressed. You may use semicolons to execute multiple commands. For example:

```
> debugbutton printxy c="print x;print y"
```

This command creates a button named **printxy** that, when clicked, prints out the values of the variables *x* and *y* in the current context.

- *iconname* is the name of the button’s icon, which may be:
 - A built-in icon. To obtain the names of built-in icons and to see what the icons look like, select **Config** → **Options** → **Editor** → **Configure Editor Buttons** or, from the Debugger, select **Config** → **Customize Toolbar** → **Add Custom Button** ().

- The filename of an icon you have created yourself. If only a partial path is given, it is assumed to be relative to the MULTI installation directory. For information about creating icons, see “Creating and Working with Icons” on page 185.
- *helpstring* is the text that appears in the Debugger window’s status bar when the cursor moves over the button.
- *tooltip* is the text that appears when the cursor hovers over the button. If you do not specify a tooltip, the name of the button is used.

The arguments *name*, *command*, *iconname*, *helpstring*, and *tooltip* must be entered as either single words or quoted strings of the form:

```
"This is a quoted string."  
"These are quotes \" \" within a quoted string."
```

You cannot save **debugbutton** changes across MULTI sessions. As a result, this command is generally only useful for the creation or modification of buttons executed by scripts. For information about how to change the Debugger toolbar permanently, see “Adding, Removing, and Rearranging Toolbar Buttons” in Appendix A, “Debugger GUI Reference” in the *MULTI: Debugging* book.

To save **editbutton** changes across MULTI sessions, select **Config → Save Configuration as User Default**.

Note: The **debugbutton** command does not affect customizations you make through the **Customize Toolbar** window. For information about this window, see “Adding, Removing, and Rearranging Toolbar Buttons” in Appendix A, “Debugger GUI Reference” in the *MULTI: Debugging* book.

The following special forms of **debugbutton** and **editbutton** can be used in the Debugger command pane:

- **debugbutton** or **editbutton**— Lists defined Debugger or Editor buttons. The **debugbutton** command does not list the **Close Debugger** button or the separator before it; these cannot be modified or deleted.
- **debugbutton 0** or **editbutton 0** — Deletes all Debugger or Editor buttons, with the exception that the **debugbutton 0** command does not delete the **Close Debugger** button or the separator before it.
- **debugbutton num** or **editbutton num** — Deletes the Debugger or Editor button numbered *num*. You can view button numbers by entering **debugbutton** or **editbutton** in the command pane.

- **debugbutton** *name* [...] — If no optional arguments are specified, deletes the button named *name*. If optional arguments are specified and if a button named *name* exists, the button is replaced. Otherwise a new button named *name* is added to the end of the Debugger toolbar. You can view button names by entering **debugbutton** in the command pane.
- **debugbutton** *num name* [...] — Replaces the button numbered *num* with a new button named *name*. You can view button numbers by entering **debugbutton** in the command pane.

Creating and Working with Icons

In addition to numerous built-in icons, MULTI can use graphic files you provide for icons. If you create your own graphic file, it must be in the Windows Bitmap format (**.bmp**). If you plan to use the bitmap in a UNIX environment, it must additionally be in an uncompressed 16- or 256-color format. An easy way to create such a bitmap is to use the Paint accessory in Microsoft Windows. Select **16 Color Bitmap** or **256 Color Bitmap** for the **Save as type** in the **Save As** dialog box.

- The built-in icons in MULTI are 16 pixels wide by 16 pixels tall, so your buttons will look best if you also use this size for your custom bitmaps.
- By default, the color light gray in your custom icons will become transparent.

On UNIX, you can specify color translations for your custom icon by appending a string of the form "oldcolor1=newcolor1&oldcolor2=newcolor2" with a question mark to the end of your bitmap filename. For example:

```
> debugbutton Hello c="echo hello" \
continued> i="/home/user/hello.bmp?black=fg&dkgray=shadow&white=highlight" \
continued> h="Say hello"
```

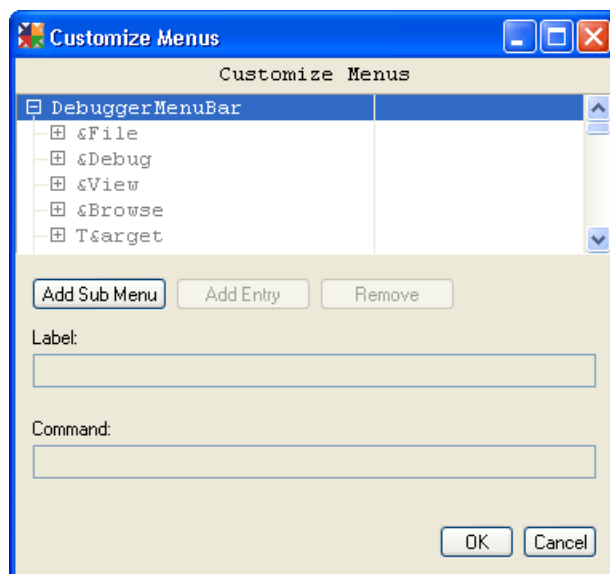
The above example would replace black pixels in **hello.bmp** with the current foreground color, dark gray pixels with the current shadow color, and white pixels with the current highlight color. You can use the following values for *oldcolor* and *newcolor*:

Oldcolor (RGB values)	Possible values for newcolor
white (255, 255, 255)	white [default] highlight

Oldcolor (RGB values)	Possible values for newcolor
ltgray (192,192,192)	transparent [default] ltgray
dkgray (128,128,128)	dkgray [default] shadow
black (0,0,0)	black [default] fg

Configuring and Customizing Menus

MULTI comes with a set of predefined menus for the Debugger, Editor, and Project Manager, but you can create new menus or add to existing menus from within any of these tools by using the **Customize Menus** window.



The **Customize Menus** window allows you to add new menus to the end of the menu bar; add new submenus and menu items to the end of existing menus; and remove menus, submenus, and menu items that you have added.

To access this window from the Debugger or Editor:

- Select **Config** → **Customize Menus**.
- Use the Debugger or Editor **customizemenus** command.

To access this window from the Project Manager:

- Select **Tools** → **Customize Menus**.

After you have opened the **Customize Menus** window, you can add a new menu to the end of a tool's menu bar:

1. Select the menu bar you want to add the menu to: the **DebuggerMenuBar**, **EditorMenuBar**, or **ProjectManagerMenuBar**.
2. Click **Add Submenu**.
3. Enter a menu name in the **Label** text field.

Adding a submenu is very similar to adding a menu: instead of selecting *tool*/**MenuBar**, select the menu you want to add the submenu to. Then follow steps 2 through 3 above. MULTI adds the submenu to the end of the menu you selected.

To add a new menu item to the end of an existing menu or submenu:

1. Select the menu or submenu you want to add to.
2. Click **Add Entry**.
3. Enter a name for your menu item in the **Label** text field.
4. In the **Command** field, enter a valid command that you want to be executed whenever you select the menu item. If you are adding the menu item to:
 - The **DebuggerMenuBar** — Enter a MULTI Debugger command. See the *MULTI: Debugging Command Reference*.
 - The **EditorMenuBar** — Enter a MULTI Editor command. See Appendix A.
 - The **ProjectManagerMenuBar** — Enter a MULTI Project Manager command. See the following table for a list of Project Manager commands.

ConnectByName connectName: " <i>connection_method_name</i> "
Connects to the Connection Method <i>connection_method_name</i> .
DebugByName debugPath: " <i>filename</i> "
Opens the executable file <i>filename</i> in the MULTI Debugger.
EditOtherByName editPath: " <i>filename</i> "
Opens <i>filename</i> in your configured editor.

ExecuteCommandOnSelected **commandSpec:** "*command*"

Executes the system command *command* on the file(s) selected in the Project Manager and directs output to the Project Manager **Status** pane. If more than one file is selected, *command* is run once for each file.

By default, the full path of the selected file is appended to the end of *command*. To include the full path elsewhere (for example, in the middle of *command*), specify *%filename* in the command string. When *command* executes, *%filename* is replaced with the full path of the selected file.

ExecuteShellCommandOnSelected **commandSpec:** "*command*"

Behaves the same as **ExecuteCommandOnSelected** (above) except that *command* is executed in a separate shell window and output is sent to the shell window instead of to the Project Manager **Status** pane.

LoadModuleByName **modulePath:** "*filename*"

Downloads the object module *filename* to your run-mode target.

OpenProjectByName **projectPath:** "*filename*"

Opens the project file *filename* in the Project Manager.

The label **local config** appears to the right of menus, submenus, or menu items you added locally. The label **site config** appears to the right of menus, submenus, or menu items that result from changes saved to a site-wide configuration directory.

Menus, submenus, and menu items that you add through the **Customize Menus** window are saved by default. To delete them, open the **Customize Menus** window, select the desired item, and click **Remove**.

The **Customize Menus** window does not allow you to rename default menus or menu items, nor does it allow you to change the command associated with a default menu item.

In addition to using the **Customize Menus** window, you can also configure menus by using the following methods:

- Enter the **menu** command along with appropriate arguments.
- Edit the **menu.odt** configuration file located in one of the following directories:
 - Windows — *user_dir*\Application Data\GHS\v5
 - UNIX — *user_dir*/.ghs/v5



Note Menu names are case-sensitive.

You can view and manipulate menus with the **menu** command.

menu

Prints a list of all the currently defined menus.

menu name

Prints the body of the menu named *name*. To view menu names, enter **menu**.

menu name { { label cmd }... }

Defines a menu that can be inserted into a menu bar or displayed by a MULTI button, mouse button, or key binding. The arguments in this command have the following meanings:

name is the menu name that appears on the menu bar or at the top of the menu when it is opened.

label is an entry in the menu.

cmd is a command that executes when the associated *label* is selected.

Menus can contain other menus by using the **->** command. For example, the menu *Main* defined below contains *RunCmds* as a submenu:

```
menu Main { {RunCmds -> RunCmds} {Up {E 1}} {Down {E -1}} {ToPC E} }
```

Note: To replace an existing menu, define a new menu with the same name.

To create a menu:

1. Enter the command **menu** followed by the *name* for the menu.
2. Create menu items by entering a *label* and an associated *cmd* enclosed in curly braces { } .
 - If one of the characters in a menu *label* is preceded with an ampersand (&), that character becomes a *hotkey* and will be underlined when the menu is displayed. The user can type that character to execute the associated command, just as if they clicked the label.
 - The *cmd* portion can contain its own subset of curly braces for commands that require them (such as **if**), but they must be paired correctly.
 - If *cmd* is a single command and that command is associated with a key binding that key binding will be displayed to the right of *label* when the menu is displayed.

3. Enclose the entire body of the menu in curly braces { }.

The following example creates a menu named RunCmds:

```
menu RunCmds {{Step s}{Next S}{Run r}{Go c}{Return cU}}
```

Opening and Accessing Menus

You can access new menus in the following ways:

-> name

Opens a menu named *name*. For example, to open the **Main** menu from the Debugger command pane, enter:

```
->Main
```

debugbutton [name]-> [menu]

Binds a menu named *menu* to the Debugger button named *name*.

For example, the following command will create a button named **View** on the Debugger toolbar, which opens the **View** menu when it is clicked:

```
debugbutton View ->ViewMenu
```

This command is most useful when used in conjunction with the **menu** command to define your own menus. For information about the **menu** command, see “Configuring and Customizing Menus” on page 186.

You cannot save **debugbutton** changes across MULTI sessions. As a result, this command is generally only useful for the creation or modification of buttons executed by scripts.

mouse [button_num] [*click_num] [@location] ==> [menu]

Binds a menu to a mouse click. For example, the following command displays the **RunCmds** menu when the third mouse button is clicked:

```
mouse mouse3*Press1@All==>RunCmds
```

For more information about binding mouse clicks to commands, see “Customizing Keys and Mouse Behavior” on page 191.

Customizing Keys and Mouse Behavior

You can customize Debugger and Editor key bindings and the behavior of mouse clicks by doing one of the following:

- Select **Config** → **Options** to open the **Options** window. Click the **General** tab, then click the **Keys** button to open the **Keyboard Commands** dialog box, or click the **Mouse** button to open the **Mouse Commands** dialog box. Make appropriate changes.
- In the Debugger command pane, enter the **keybind** or **mouse** command with appropriate options. For information about these commands, see the following sections.

Customizing Keys with the **keybind** Command

keybind

keybind *location*

keybind *key* [*modifiers*] [*@location*] [=*command*]

This command binds a key to a command or displays key bindings, depending on the options given.

Entering this command with no arguments (the first command shown) displays key bindings that apply in all contexts (*location* specified as **All**). For information about *location*, see “Key and Mouse Locations” on page 195.

Entering the second command displays the current key bindings for a particular *location*. See “Key and Mouse Locations” on page 195.

The third command assigns an action to take when a key is pressed in the specified context. The arguments for this command have the following meanings:

- *key* is an identifier that specifies the keyboard key for which binding is active. This identifier may be either a single ASCII (or ISO8859) character or a quoted string containing the name of one of the keys on the keyboard, such as “BACKSPACE” or “F3.” Characters needing more than one key press (besides the Shift, Ctrl, Alt, and Meta keys) cannot be used.

- To obtain a list of all of the acceptable key names, type `keybind "?????"`.
- The BACKSPACE key is different from `h|Control` even though their ASCII representations are the same.
- To specify the double quotation mark key, use a sequence of three double quotation marks in a row (`" " "`).
- *modifiers* are other keys (Shift, Ctrl, Alt, and Meta) that may be used in combination with *key*. If a modifier is specified, the command runs only if that modifier is pressed at the same time as the *key*. If more than one modifier is specified, they should be separated from each other and from *key* by vertical bars (`|`).
- *location* is the area of the window the cursor must be in for the key binding to be matched. If *location* is omitted, the default is `Command`. For more information, see “Key and Mouse Locations” on page 195.
- *command* is the command that will be executed upon the key press. For more information, see “Key and Mouse Command Special Sequences” on page 196.

If the `key[modifier]@location` portion of the **keybind** command matches a previously defined binding, then the new definition replaces the old binding. If no command is specified in the new definition, then the old binding is deleted.

When a key is pressed in a window, MULTI searches for key bindings that match. There may be more than one, in which case MULTI chooses the one whose location is the most specific. If there are still several, then MULTI chooses one arbitrarily.

Example 2. keybind Command

With this command, MULTI evaluates and prints the selection when the first function key (F1) is pressed in the source pane or command pane:

```
keybind "F1"=print %s
```

With this command, the Debugger opens a Data Explorer that displays the current selection when the key combination Ctrl+F1 is pressed:

```
keybind "F1"|Control@All=view %s
```


With these commands, MULTI scrolls the window while the up or down arrow is pressed:

```
keybind "Up"@All=(%w) scrollcommand 11
keybind "Down"@All=(%w) scrollcommand -11
```

With this command, MULTI single-steps the process when Ctrl+S is pressed in the source window:

```
keybind s|Control=S
```

Customizing Mouse Behavior with the mouse Command

mouse

mouse *location*

mouse *button_num* [**Clickclick_num*[(AtOnce)]] [*modifiers*] [*@location*]
[*=command*]

This command defines the function of a mouse button or displays mouse bindings, depending on the options given.

Entering this command with no arguments (the first command shown) displays all current mouse bindings.

Entering the second command displays the current mouse bindings for a particular *location*. For information about *location*, see “Key and Mouse Locations” on page 195.

The third command assigns an action to a mouse button click. The arguments for this command have the following meanings:

- *button_num* represents the mouse button to be assigned. This can be the keyword `Any`, which matches any mouse button, or the word `mouse`, followed by a combination of digits between 1 and 5 which match the specified mouse button. For example, `mouse2` matches the second (usually the middle) mouse button, while `mouse13` matches both the first (usually the left) and the third (usually the right) buttons. Not all mice have five buttons. Commands assigned to non-existent buttons will never be matched.
- The keyword `Click` may be replaced by `Press`, which executes the command on the button press rather than the release, or by `Either`, which

executes the command on both the press and release. A binding that contains `*Either1` is executed twice, once for the press of the mouse button, and once for the release.

- *click_num* specifies the number of times the mouse button must be released before the command is executed. This may be a number between 1 and 5. A binding that contains `*Click3` would be executed upon the release of the specified button on a triple click. If the `*Clickclick_num` option is omitted, the binding matches the first release of the specified mouse button.
- The keyword `(AtOnce)` causes the command assigned to execute immediately rather than pausing to see if it is part of a click sequence. A binding that contains `*Press2` is executed only after the second press of a mouse button in a double-click, while `*Press2 (AtOnce)` would be executed for the second mouse button press in a double-click, triple-click, quadruple-click, etc. This is acceptable behavior for many options, such as the standard selection clicks: one click sets the insertion point, two clicks selects the current word, three the line, and so forth. If `(AtOnce)` is not used, there is a slight delay when invoking single-click commands. This delay is specified by the **ClickPause** configuration option.
- *modifiers* are other keys that must be held down at the same time the button is clicked. Valid *modifiers* are Shift and Ctrl (plus Meta for UNIX). Modifiers are preceded by a vertical bar (`|`), which separates them from each other and from the previous arguments. If more than one modifier is specified, all of the modifiers listed must be pressed simultaneously with the mouse click for the binding to be matched.
- *location* indicates the place within MULTI's windows the mouse must be for the binding to be matched. If *location* is omitted, the default is `Source`. For more information, see "Key and Mouse Locations" on page 195.
- *command* is the MULTI command that will be executed when the binding is matched. For more information, see "Key and Mouse Command Special Sequences" on page 196.

If the *button_num*`[*Clickclick_num][|modifiers][@location]` portion of a mouse binding matches a previously defined mouse binding, then the new binding replaces the old one. If no *command* is specified, then the old mouse binding is deleted.

When the mouse is clicked in a window, MULTI searches its list of mouse actions for a matching binding. If there is more than one that might match the event, then MULTI uses the one whose location is the most specific. If there are

multiple bindings that match the most specific location, then MULTI chooses the one which is bound to the fewest number of different mouse buttons. If there are still several bindings, then MULTI chooses one of the bindings arbitrarily.

Example 3. Mouse Command

With this binding, the Debugger evaluates and prints the selection when you click the left mouse button once in the source window.

```
mouse mouse1=print %s
```

With this binding, the Debugger opens a Data Explorer displaying the current selection when you double-click the left mouse button in any window.

```
mouse mouse1*Click2@All=view %s
```

Key and Mouse Locations

The *location* argument for the **keybind** and **mouse** commands can be one of the following:

All	keybind — Any MULTI window except the Editor. mouse — Any MULTI window.
InputWindow	Any MULTI input interface such as the command pane.
OutputWindow	Any output-only window, such as the source pane or a monitor window.
Command	keybind — The command pane or source pane. mouse — The command pane.
Source	mouse only — The source pane.
Monitor	Anywhere in a monitor window except the title bar.
Freeze	Anywhere in the region indicating whether or not the window is frozen.
Close	The close button in the title bar.
ScrollBar	Anywhere within a scroll bar.
UpArrow	keybind only — Arrow pointing up at the top of a scroll bar.
DownArrow	keybind only — Arrow pointing down at the bottom of a scroll bar.
ScrollArea	keybind only — Region between the direction arrows of a scroll bar.

Thumb	keybind only — The control in the middle of a scroll bar that can be dragged to change the viewable contents of the associated window.
AboveThumb	keybind only — Area between the Thumb and the UpArrow.
BelowThumb	keybind only — Area between the Thumb and the DownArrow.
Edit	Anywhere within an Editor.

Some of these locations implicitly include other locations. These locations are:

All	InputWindow, OutputWindow, and View
InputWindow	Command
OutputWindow	Source and Monitor
ScrollBar	UpArrow, DownArrow, and ScrollArea
ScrollArea	keybind only — Thumb, AboveThumb, and BelowThumb

Key and Mouse Command Special Sequences

The *command* is any MULTI command to be executed when the key or mouse binding is matched. Several special sequences of characters in the *command* will be replaced with information about MULTI's state when the action is performed. These sequences are not valid for key or mouse bindings in the Editor. The following are the special sequences for *command*:

%s	Replaced by the current selection.
%p	Replaced by the current selection if it exists, otherwise MULTI prompts for input.
%P	Always prompts for input.
%w	Replaced by a special number identifying the window to MULTI. In command synopses and MULTI documentation, this number is referred to as a <i>window identification number</i> or <i>wid</i> .
%x	Replaced by the <i>x</i> location in the window when pressing the button or mouse.
%y	Replaced by the <i>y</i> location in the window when pressing the button or mouse.
%k	(keybind) — Replaced by the ASCII expansion of the key. For the key labeled <i>a</i> , this is replaced with "a". (mouse) — Replaced by the null string.

%m	(keybind) — Replaced by <code>Press</code> to indicate a key press triggered the action. (mouse) — Replaced by <code>Press</code> or <code>Release</code> , depending on whether the command was executed as a result of a mouse button press or a mouse button release.
%%	Replaced by %.

Inserting a Character Blocked By a Custom Key Binding

If you customize the Editor to run a command based on a single keystroke, you will not be able to directly insert the literal character of that keystroke into a file. For example, if you customize the key binding so that every time you press `d` the cursor moves down one line, then you will not be able to type the literal letter `d` in a file. To enter the literal character `d` in your file, you would have to:

1. Press `Ctrl+\` (the `Ctrl` key with the backslash “`\`” key).
2. Press the key for the character that you want to enter into the file.

Configuring Taskbar Organization

On some versions of the Windows operating system, **MULTI** windows are grouped under a single taskbar entry to free up the taskbar. You can configure **MULTI** to group windows under a system tray icon instead, or you can disable the taskbar organization feature altogether. Clicking either the taskbar entry or system tray icon provides access to all **MULTI** windows via a shortcut menu (described in the next section).

This feature is especially useful if you open large numbers of **MULTI** windows while you work.



Note Taskbar organization is supported on Windows 2000, XP, and Vista for 32-bit processors. It is not supported on Vista for 64-bit processors or on UNIX.

The following bullet points describe the two taskbar organization modes:

- **Taskbar mode** — [default] Adds an entry labeled **MULTI** to the Windows taskbar, and groups all **MULTI** windows under this entry.

- Tray icon mode — Adds a MULTI icon () to the system tray—the taskbar status area located at the far right side of the taskbar—and groups all MULTI windows under this icon. To activate the system tray icon, you can click it or do the following (Windows XP and Vista only):

1. Press the Windows key + B.
2. Use the arrow keys to navigate the tray icons.
3. Press Enter when the desired icon is selected.

To configure taskbar organization, perform the following steps:

1. Select **Config** → **Options** from a major MULTI tool.
2. In the **Options** window that appears, click the **General** tab.
3. Click the **Windows** button located at the bottom of the tab.
4. Select **Taskbar Organizer** and click **OK**.
5. In the **Taskbar Organizer** dialog box that appears, adjust your configuration settings as desired. For detailed information about the configuration settings, see “The Taskbar Organizer Dialog Box” on page 216.
6. Click **OK**.



Note If the Taskbar Organizer is enabled (in either taskbar mode or tray icon mode), the Windows key + M key combination does not minimize MULTI windows. The Windows key + D key combination and the **Show Desktop** button both hide all windows, but MULTI windows may reappear when another application is brought to the front. You can disable the Taskbar Organizer by following the preceding configuration instructions.

Taskbar Organizer Menu Options

The menu that appears when you click the **MULTI** taskbar entry or  system tray icon allows you to display any window that is currently open in the MULTI IDE. The most recently used windows are displayed at the top of the menu, and all other windows are grouped into submenus according to their type. For example, at the top of the menu, you might see the recently accessed Debugger program *hello* followed by the Editor file *hello.c*. Under the separator bar, you would see your Debugger windows grouped under the **Debuggers** submenu and MULTI Editor windows grouped under the **Editors** submenu.

The menu also contains the following menu items:

- **Settings** — Opens the **Taskbar Organizer** dialog box. For detailed information about the configuration settings located in this dialog box, see “The Taskbar Organizer Dialog Box” on page 216.
- **Show All** — Displays MULTI IDE windows that were previously minimized.
- **Minimize All** — Minimizes all MULTI IDE windows.
- **Exit All** — Exits all running MULTI IDE processes. A dialog box asking you to confirm this action appears.



Note In taskbar mode, clicking the **MULTI** taskbar entry when only a single window is open brings that window to the front. To access the menu while in taskbar mode, more than one window must be open.

Configuring Window Docking

MULTI includes configurable *window docking* options, which allow you to customize how multiple windows interact and are displayed on the screen. Window docking is supported in both Windows and UNIX environments. For information that is specific to UNIX environments, see “UNIX Docking Limitations” on page 200.

The following two types of behavior are supported:

- **Window alignment** — Windows of the same application automatically snap together when they are brought near each other, and windows brought near the edge of your screen automatically snap to the edge of the screen. You can configure the distance at which a window snaps to another window or screen border. See **Docking Distance** in “Window Docking Global Options” on page 218.
- **Window grouping** — Windows of a similar type form groups when they are brought near each other. You can move, close, or minimize the entire group of windows by using the group title bar. Do not use the **Maximize** button on the group bar.



Tip To see or configure the window types that group together, perform the following steps:

1. Select **Config** → **Options**.
2. Click the **General** tab.
3. Click the **Windows** button located at the bottom of the dialog box.
4. Windows only — Select **Window Docking** and click **OK**.
5. Click the **Application Options** tab.

For information about window docking options, see “The Window Docking Options Window” on page 218.

UNIX Docking Limitations

Due to the design of the X protocol, the window manager running has nearly absolute control over window position and size, and the application is reduced to a much less active role. Window docking attempts to give you the benefits of a window manager at the application level.

Some window managers, such as KDE, do not allow a window to be created smaller than a certain size. While you can still use window groups in these situations, the group bar takes up more space than usual.

Unless otherwise stated, the following X Window Managers support both window alignment and window grouping.

- afterSTEP
- Blackbox
- Desktop Window Manager (**dtwm**)
- FVWM, FVWM2, FVWM95
- Ice Window Manager (**icewm**)
- KDE Window Manager (**kwin**) — Window grouping is disabled by default because the KDE window manager does not allow windows to be created with a height smaller than 25 pixels. You can still use window grouping; however, the group bar takes up more space than usual.
- Metacity

- OpenLook Window Manager (**olwm**)
- Sawfish
- Tabbed Window Manager (**twm**)
- WindowMaker (**wmaker**)
- wm2 — Window grouping is not supported because wm2 windows do not use a horizontal title bar, which is required for window grouping.

The following X Window Managers do not work properly with window docking and are not supported. If you use one of these window managers, set Window Docking Options to **Disable All Window Management** (see “Window Docking Global Options” on page 218).

- PWM
- Ion

If your window manager is not named in either of the preceding lists, it has not been verified to work properly.

Configuring Window Focus

When MULTI launches a new window or dialog box, it is brought to the foreground and given input focus even when MULTI is not the current application. The appearance of some MULTI dialog boxes, such as the one indicating that an executable has been rebuilt, may interrupt a previous foreground application by taking focus away from it.

On Windows, perform the following steps to help prevent this:

1. Download and install Tweak UI from the Microsoft Web site (www.microsoft.com). Note that Tweak UI is not available for Windows Vista.
2. Launch the **Tweak UI** window:
 - Windows XP — Select **Tweak UI** from the Start menu.
 - Earlier Windows versions — Select **Tweak UI** from the **Control Panel**.
3. In the **Tweak UI** window, expand **General** and select **Focus**.
4. Select **Prevent applications from stealing focus**.

5. Click **OK**.

On UNIX, window managers are solely responsible for maintaining window focus. Some window managers, such as KDE, have configurable window focus settings. However, adjusting such settings may prevent MULTI from bringing its windows to the foreground even when MULTI is the current application.

Configuring File Extensions

When you are using the MULTI IDE, some operations require you to select a file from a graphical file chooser. These file choosers provide a set of filters that allow only the relevant files to be displayed. While common extensions are present in these filters, additional custom extensions and file types may be required. These extensions can be added by customizing the file extension mappings used by MULTI.

Extension Mapping Files

The extension mappings are stored in configuration files named **extensions.udb** in a directory named **file_types**. They are applied in the same order as configuration files, starting with the **defaults** directory and followed by the site-wide and user configuration directories (see “Creating and Editing Configuration Files” on page 175).

To add custom file extensions, create a directory named **file_types** in either the site or user configuration directories and create an **extensions.udb** file defining the custom file extension. The extension mapping configuration files contain entries describing individual file types as well as collections of file types for use in individual file choosers.

For an example **extensions.udb** file, see:

install_dir/defaults/file_types/extensions.udb

This file contains the default extension mappings.

See also the **fileextensions** command in “General Configuration Commands” in Chapter 7, “Configuration Command Reference” in the *MULTI: Debugging Command Reference* book.

Configuring File Associations (Windows only)

When you install MULTI on Windows, the installer prompts you to choose what program you would like to use to open files of various extensions. You may choose to associate certain file types with the MULTI Project Manager and certain file types with the MULTI Editor. Perform the following steps to change these file associations at a later time.

If you are using Windows 2000/XP:

1. Access the **Control Panel's Folder Options**.
2. Click the **File Types** tab, and make the desired changes.

If you are using Vista:

1. Access the **Control Panel's Default Programs**.
2. Click **Associate a file type or protocol with a specific program**, and make the desired changes.

Configuration Options

Chapter 9

This Chapter Contains:

- General Configuration Options
- Project Manager Configuration Options
- Debugger Configuration Options
- MULTI Editor Configuration Options
- Session Configuration Options
- Colors Configuration Options
- Deprecated Configuration Options

Chapter 8, “Configuring and Customizing MULTI” discusses how you can customize your interface using MULTI’s configuration options. This chapter describes these configuration options and their default settings.

You can set many, but not all, of the configuration options via the **Options** window. To open the **Options** window:

- From the Launcher, Debugger, or Editor, select **Config** → **Options**.
- From the Project Manager, select **Tools** → **Options**.

You can also set or edit all of the options available in the **Options** window (and more besides) by entering the following command in the Debugger command pane:

```
configure config_option [:|=] value
```

For options that cannot be set in the **Options** window, *config_option* and possible *values* are listed at the beginning of the table row dedicated to the option. For options that can be set in the **Options** window, *config_option* and *values* are listed—often in parentheses—in the option description. For example, in this chapter, the GUI option **Match exact case in searches** is followed by two option/value pairs in parentheses:

- **Selected (exactCase on)**
- **Cleared (exactCase off)**

This indicates that you could either enter:

```
configure exactCase on
```

or

```
configure exactCase off
```

in the Debugger command pane. When you modify configuration settings via the **configure** command, the modifications are not saved by default. For more information, see “Using the configure Command” on page 171. In general, if you change the setting of a configuration option without saving the modification, only the MULTI tool from which the option was changed is aware of the new setting. For more information, see “Propagating Configuration Settings” on page 174.

Options that can be set via the **configure** command can also be set in configuration files. Syntactically, the only difference between entering an option in a configuration file and entering it in the command pane is that you do not include **configure** in the configuration file. For information about setting configuration options in a configuration file, see “Creating and Editing Configuration Files” on page 175.



Note Configuration options are case-insensitive; thus `exactCase off` and `exactcase off` are equivalent and are both valid. In this chapter, we use mixed-case notation where it makes the options more readable.



Tip Entering `help config_option` in the Debugger command pane opens online help for the specified configuration option.

General Configuration Options

This section describes configuration options that affect the overall appearance of MULTI. Most of these options appear on the **General** tab of the **Options** window.

The General Options Tab

To access the following options, select **Config** → **Options** → **General** tab.

For information about setting configuration options from the command pane, see the introduction of Chapter 9, “Configuration Options”.

Save window positions and sizes

Permitted settings for this option are:

- **Selected (rememberWindowPositions on)** — [default] Remembers the position and size of windows created by MULTI so that the next time a window of the same type is created, it appears in the same location and with the same size. For example, if you resize the Project Manager window, move it to a specific location on your screen, and then close the window, the next Project Manager you open appears in the same place and with the same size. The first window of a given type is positioned in the saved location. Any new windows of the same type are slightly offset from the location of the open window and are given the saved size. Size and position are remembered even after you exit and restart MULTI.
- **Cleared (rememberWindowPositions off)** — Does not remember the position and size of windows created by MULTI.

Note: This setting does not apply to certain types of dialog boxes, such as modal dialog boxes, or certain windows, such as the **Active Licenses** window, the **Memory Test Wizard**, the **Perform Memory Test** window, and the **Source Path** window.

Display close (x) buttons

Permitted settings for this option are:

- **Selected (closeButtonOnTitlebar on)** — A **Close** button will appear on window toolbars. This option only affects new windows created after you change this setting, except for Debugger windows, which adjust to the new setting immediately.
- **Cleared (closeButtonOnTitlebar off)** — [default] The **Close** button will not appear on toolbars.

Launch clipboard manager

UNIX only

Permitted settings for this option are:

- **Selected (clipManLaunch on)** — [default] The clipboard manager, which stores selections that have been sent to the clipboard by all applications, will run after MULTI has exited, making the contents of the clipboard accessible to other applications.
- **Cleared (clipManLaunch off)** — The contents of the clipboard are lost when MULTI exits.

Match exact case in searches

Determines the case sensitivity of interactive text searches in the Editor and of incremental text searches in MULTI tools. Permitted settings for this option are:

- **Selected (exactCase on)** — Searches are case-sensitive.
- **Cleared (exactCase off)** — [default] Searches are case-insensitive.

This option can also be set on a per-Editor-window basis with the Search dialog box. For more information, see “Interactive Searching Using the Search Dialog Box” on page 29.

Note: This setting does not apply to the **grep** Editor or Debugger commands or to any of the Search in Files dialog boxes, all of which run **grep** and are case-sensitive by default.

Escape restores view after iSearch

Permitted settings for this option are:

- **Selected (iSearchReturn on)** — Returns the cursor to its original position when you press Esc while in incremental search mode. See “Incremental Searching” on page 28.
- **Cleared (iSearchReturn off)** — [default] Does not return the cursor to its original position when you press Esc while in incremental search mode.

Allow beeping

Permitted settings for this option are:

- **Selected (beep on)** — [default] MULTI beeps on various error conditions, such as a search that does not match anything.
- **Cleared (beep off)** — MULTI never beeps.

For information about the **Beep** Editor command, see “Insert Commands” on page 293.

Show tooltips

UNIX only

Permitted settings for this option are:

- **Selected (tooltips on)** — [default] Tooltips (small explanatory boxes that pop up when you hover the cursor over a GUI item) will appear if available.
- **Cleared (tooltips off)** — Tooltips will never appear.

Warp pointer

Permitted settings for this option are:

- **Never (warpPointer never)** — [default for Windows] Never move the mouse pointer automatically. The mouse cursor can only be changed by physically moving the mouse.
- **Into Dialog Box (warpPointer intoDialogue)** — [default for UNIX] Warp the mouse pointer to the default button of new dialog boxes that appear.
- **In & Out of Dialog Box (warpPointer in&OutDialogue)** — Warp the mouse pointer into the default button of new dialog boxes, and warp it back to its previous location when the dialog box is closed.

Print command

UNIX only

(**printCommand** *command*) Specifies the command used to send information to a printer. The default is **lpr**.

Editor

Specifies which editor MULTI uses when you request an editor to start. Permitted settings for this option are:

- **MULTI Editor (extEditor_Choice "multi editor")** — [default] Uses the MULTI Editor.
- **Emacs (extEditor_Choice emacs)** — Uses the Emacs Editor.
- **Vi (extEditor_Choice vi)** — Uses the vi Editor.
- **Other (extEditor_Choice other)** — Uses an editor that is not available in the list.

Configure Editor

Opens a dialog box that allows you to specify configuration settings specific to the editor selected in the **Editor** field. For more information, see “Third-Party Editor Configuration Options” on page 213.

If **MULTI Editor** is selected in the **Editor** field, clicking this button opens the **MULTI Editor** tab. For more information, see “The MULTI Editor Options Tab” on page 251.

Version Control

Specifies which version control system MULTI uses. Permitted settings for this option are:

- **Auto Detect** (**versionControlType autoDetect**) — [default] Version control is auto-detected based on the file currently in use.
- **Disable** (**versionControlType disable**) — Version control is disabled.
- **CVS** (Concurrent Versions System) (**versionControlType cvs**) — An open-source network-transport version control system. See “Integrating with CVS” on page 129.
- **Subversion** (**versionControlType subversion**) — An open-source network-transport version control system. See “Integrating with Subversion” on page 134.
- **RCS** (**versionControlType rcs**) — A version control system available with many UNIX systems. See “Integrating with RCS” on page 131.
- **ClearCase** (**versionControlType clearCase**) — A version control system from Rational Software. See “Integrating with ClearCase” on page 128.
- **SourceSafe** (**versionControlType sourceSafe**) — (Windows only) A version control system from Microsoft. See “Integrating with SourceSafe” on page 132.
- **PVCS** (**versionControlType pvcs**) — (Windows only) A system of version control and configuration management software from MERANT. See “Integrating with PVCS” on page 130.

Configure Version Control

Opens a dialog box that you use to specify configuration settings that are specific to the version control selected in the **Version Control** field. For more information, see “Version Control Configuration Options” on page 215.

Phase of moon in scroll bar box

We at Green Hills decry the unfortunate tendency in our society whereby mankind perceives itself as disconnected from its environment. We as individuals pay no attention to flowers blooming or leaves falling. We live immersed in our air conditioned offices and, if we are programmers, we never see the light of day. In a perhaps futile effort to stop this trend, we provide all MULTI users with the current phase of the moon to encourage them to leave their desks and look up at the beautiful night sky.

Permitted settings for this option are:

- **Selected** (**moon on**) — Displays the approximate phase of the moon in the nook of the vertical and horizontal scroll bars.
- **Cleared** (**moon off**) — [default] The phase of the moon is not displayed.

Changes to this setting only affect new windows.

Source Code Font

(**font font_name**) Opens a font selection dialog box in which you can set the source code font. This font is used for most textual display, including in the Debugger source pane, the command pane, and the MULTI Editor. Click **Source Code Font** and select a font installed on your system. The font you choose should normally be a fixed-width font so that characters align properly. Italic and oblique fonts should not be chosen. While some italic and oblique fonts work correctly, others exhibit drawing problems.

The default font on Windows is **Courier New Regular**. The command to restore the default Windows font from the command pane is:

configure font "courier new:13"

The default font on UNIX is **misc fixed medium semi-condensed**. The command to restore the default UNIX font from the command pane is:

configure font -misc-fixed-medium-r-semicondensed-*-120-*-*-*-*-*

All MULTI tools opened during the current MULTI session are notified of changes to this option. For more information, see "Propagating Configuration Settings" on page 174.

GUI Font

(**guiFont font_name**) Opens a font selection dialog box in which you can set the GUI font. This font is used for buttons, menus, and other GUI controls. Click **GUI Font** and select a font installed on your system. Italic and oblique fonts should not be chosen. While some italic and oblique fonts work correctly, others exhibit drawing problems.

The default font on Windows is **MS Sans Serif Regular**. The command to restore the default Windows font from the command pane is:

configure guiFont "ms sans serif"

The default font on UNIX is **adobe helvetica medium**. The command to restore the default UNIX font from the command pane is:

configure guiFont *-helvetica-medium-r-normal-*-12-*-*-*-*-*

All MULTI tools opened during the current MULTI session are notified of changes to this option. For more information, see "Propagating Configuration Settings" on page 174.

Kanji Font

UNIX only

(**kanjiFont font_name**) Opens a font selection dialog which you can use to set the 16-bit font used to display Kanji text. Click **Kanji Font** and use the dialog box that opens to select any font installed on your system. There is no default setting.

All MULTI tools opened during the current MULTI session are notified of changes to this option. For more information, see “Propagating Configuration Settings” on page 174.

Menus

Opens the **Customize Menus** window, which allows you to configure menus in a number of MULTI tools. For information about how to use this window, see “Configuring and Customizing Menus” on page 186.

Mouse

Opens the **Mouse Commands** dialog box, which you can use to edit the mouse bindings used by MULTI. The mouse bindings are displayed in the same format as the **mouse** command. See “Customizing Keys and Mouse Behavior” on page 191.

Keys

Opens the **Keyboard Commands** dialog box, which you can use to edit the key bindings used by MULTI. The key bindings are displayed in the same format as the **keybind** command. See “Customizing Keys and Mouse Behavior” on page 191.

Windows

Opens a dialog box with the following options:

- **Taskbar Organizer** — (Windows only) Select to configure taskbar organization. For more information, see “The Taskbar Organizer Dialog Box” on page 216.
- **Window Docking** — Select to configure window alignment and grouping. For more information, see “The Window Docking Options Window” on page 218.

Third-Party Editor Configuration Options

To access the following options, select **Config** → **Options** → **General** tab. Then select an alternative editor (any editor other than the MULTI Editor) from the **Editor** drop-down list and click the **Configure Editor** button. (If you select **MULTI Editor** from the **Editor** drop-down list, the **MULTI Editor** tab opens. See “The MULTI Editor Options Tab” on page 251.)

The options that appear are specific to the editor type selected. Not all options appear for every editor type.

For information about setting configuration options from the command pane, see the introduction of Chapter 9, “Configuration Options”.

Path to Editor

(**extEditor_EmacsPath editor_path** for Emacs, **extEditor_ViPath editor_path** for vi, **extEditor_OtherPath editor_path** for other editors) Specifies where the editor executable is located in the system. Depending upon your `PATH` environment variable, you may need to specify the full path to the editor in the **Path to Editor** field.

Use command window

Windows only

(**extEditor_EmacsUseCmd** for Emacs, **extEditor_ViUseCmd** for vi, **extEditor_OtherUseCmd** for other editors) Determines whether the editor will be launched from within a command window. Users with console mode editors should enable this option. This is equivalent to prefixing **cmd.exe /k** to the command to launch the editor. You must restart MULTI for this change to take effect.

Use XTerm

UNIX only

(**extEditor_EmacsUseXTerm** for Emacs, **extEditor_ViUseXTerm** for vi, **extEditor_OtherUseXTerm** for other editors) Determines whether the external editor will be launched from within an xterm. Users with console mode editors should enable this option. This is equivalent to prefixing **xterm -e** to the command to launch the editor. You must restart MULTI for this change to take effect.

Command line arguments

("extEditor_EmacsArgumentFormat +%LINE %FILE0 %FILES" for Emacs, "extEditor_ViArgumentFormat +%LINE %FILE0 %FILES" for vi, "extEditor_OtherArgumentFormat *command line specification*" for other editors) Determines the arguments given to the editor when it is launched.

Three special character sequences are replaced when the editor is run. Substitution rules follow:

- **%LINE** — Replaced with the line number to go to when opening the file (removed if MULTI does not specify a line number).
- **%FILE0** (the last character is a zero) — Replaced with the first file to be edited.
- **%FILES** — Replaced with all but the first file if there is more than one file to edit (removed if MULTI only specifies one file to edit). Each file is separated by a space.

Version Control Configuration Options

To access the following options, select **Config** → **Options** → **General** tab. Then click the **Configure Version Control** button. (If you select **Custom** as your version control type, a **Custom Configuration** dialog box appears. See .)

The options that appear are specific to the version control type selected. Not all options appear for every version control type.

For information about setting configuration options from the command pane, see the introduction of Chapter 9, “Configuration Options”.

Automatic Checkout

Permitted settings for this option are:

- **Selected** (**versCtrl_RcsAutoCheckout on** for RCS, **versCtrl_ClearCaseAutoCheckout on** for ClearCase, **versCtrl_SourceSafeAutoCheckout on** for SourceSafe, **versCtrl_PvcsAutoCheckout on** for PVCS) — Files are automatically checked out when you start to edit them.
- **Cleared** (**versCtrl_RcsAutoCheckout off** for RCS, **versCtrl_ClearCaseAutoCheckout off** for ClearCase, **versCtrl_SourceSafeAutoCheckout off** for SourceSafe, **versCtrl_PvcsAutoCheckout off** for PVCS) — [default] Files are not automatically checked out.

This option is not available if you have selected the CVS or Subversion version control systems because a file is always checked out when you use CVS or Subversion.

Location of VC binary

For CVS, ClearCase, SourceSafe, and Subversion version control systems only.

Specifies the name of the command to run, according to your version control system. The defaults are:

- **cvs** (**versCtrl_CvsPath cvs**) — For CVS
- **svn** (**versCtrl_SubversionPath svn**) — For Subversion
- **cleartool** (**versCtrl_ClearCasePath cleartool**) — For ClearCase
- **ss** (**versCtrl_SourceSafePath ss**) — For SourceSafe

SourceSafe database location

SourceSafe only

(**versCtrl_SourceSafeDatabase location**) Specifies the location of the SourceSafe database. In other words, this is the location of the **srcsafe.ini** file corresponding to the desired database.

Root of checkout

SourceSafe only

(**versCtrl_SourceSafeRoot location**) Specifies the working directory of the SourceSafe database root (\$/). The MULTI integration with SourceSafe assumes that the directory structure of a database is maintained when files are checked out from that database. For example, the **\$/example.c** file in the database is checked out to **root_of_checkout/example.c**.

The Taskbar Organizer Dialog Box

The **Taskbar Organizer** dialog box allows you to free up the Windows taskbar by grouping MULTI windows into a single taskbar entry or system tray icon. You can also disable taskbar organization via this dialog box.



Note The Taskbar Organizer is only available on Windows. No equivalent feature exists for UNIX.

To access the **Taskbar Organizer** dialog box, select **Config** → **Options** → **General** tab and click the **Windows** button. Select **Taskbar Organizer** and click **OK**. The following table describes the modes and options available in

the dialog box. Note that not all options appear for every mode and that not all combinations of taskbar options are valid. The dialog box prevents you from selecting invalid combinations by dimming radio buttons that you cannot change given the current combination of settings.

Mode

Permitted settings for this option are:

- **Disabled**— Disables the Taskbar Organizer. In this mode, MULTI windows populate both the taskbar and the window list that appears when you press the Alt + Tab key combination.
- **Taskbar** — Enables taskbar mode. In taskbar mode, all MULTI windows are grouped into a single taskbar entry labeled **MULTI**. For more information, see “Configuring Taskbar Organization” on page 197.
- **Tray** — Enables tray icon mode. In tray icon mode, all MULTI windows are grouped under a single system tray icon. For more information, see “Configuring Taskbar Organization” on page 197.

When pressing Alt-Tab, display

Allows you to configure the appearance and behavior of items in the Alt + Tab list.

Permitted settings for this option are:

- **All windows** — Displays all MULTI windows in the Alt + Tab list.
- **Only the Taskbar Organizer** — Displays a single MULTI icon in the Alt + Tab list, regardless of the number of open MULTI windows. The setting of the option **When selecting the organizer in the Alt-Tab list** determines the behavior of the MULTI icon when selected.
- **Nothing** — Does not display any MULTI windows or icons in the Alt + Tab list.

When selecting the organizer in the Alt-Tab list

Determines what occurs after you select the MULTI icon in the Alt + Tab list.

Permitted settings for this option are:

- **Go to the last activated window** — Returns focus to the last active MULTI window.
- **Show a list of all windows** — Displays a shortcut menu providing access to all open MULTI windows. For more information, see “Taskbar Organizer Menu Options” on page 198.

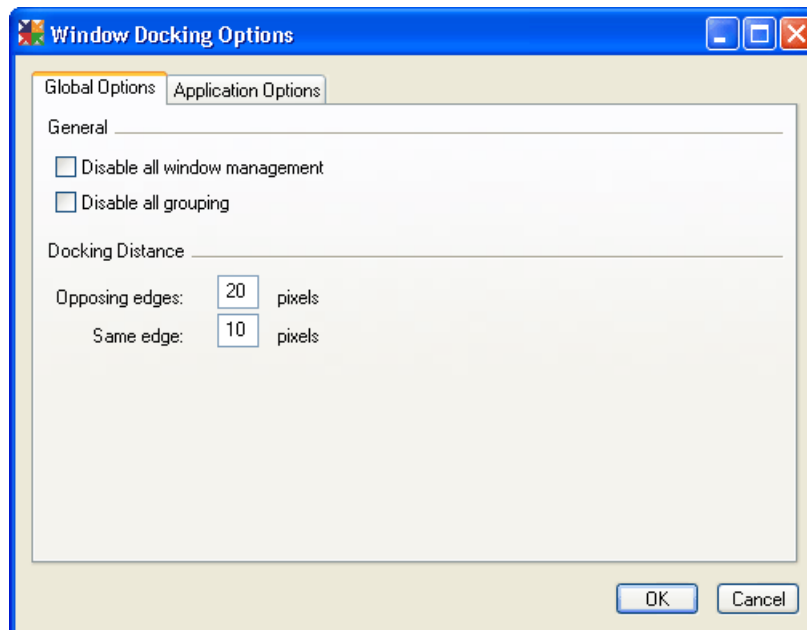
The Window Docking Options Window

To access the **Window Docking Options** window, select **Config** → **Options** → **General** tab and click the **Windows** button. On Windows hosts, select **Window Docking** and click **OK**. The resulting window contains two tabs:

- **Global Options** — The options on this tab apply to all MULTI applications (see “Window Docking Global Options” on page 218). Options selected on this tab override any application-level options set on the **Application Options** tab.
- **Application Options** — The options on this tab only apply to the application configuration file specified in the **Application** text box of the tab (see “Window Docking Application Options” on page 220). These application-level options can be overridden by options set on the **Global Options** tab.

The options on each of these tabs are described in the following sections. For more information about window docking, see “Configuring Window Docking” on page 199.

Window Docking Global Options



The items on the **Global Options** tab apply to all applications.

Disable all window management

Completely disables both window alignment (snapping) and window grouping.

Disable all grouping

Completely disables window grouping, overriding any **Application Options**.

Docking Distance

Specifies the distance at which windows snap to each other, either side by side or vertically. There are two settings:

- **Opposing edges** — Specifies the distance (in pixels) at which windows snap together when the left edge of one window snaps to the right edge of another, or the top of one window snaps to the bottom of another. The default is 20.
- **Same edge** — Specifies the distance (in pixels) at which windows snap together when you align the edges of vertically adjacent windows. For example, you can align a series of windows in a column along the left edge. The default is 10.

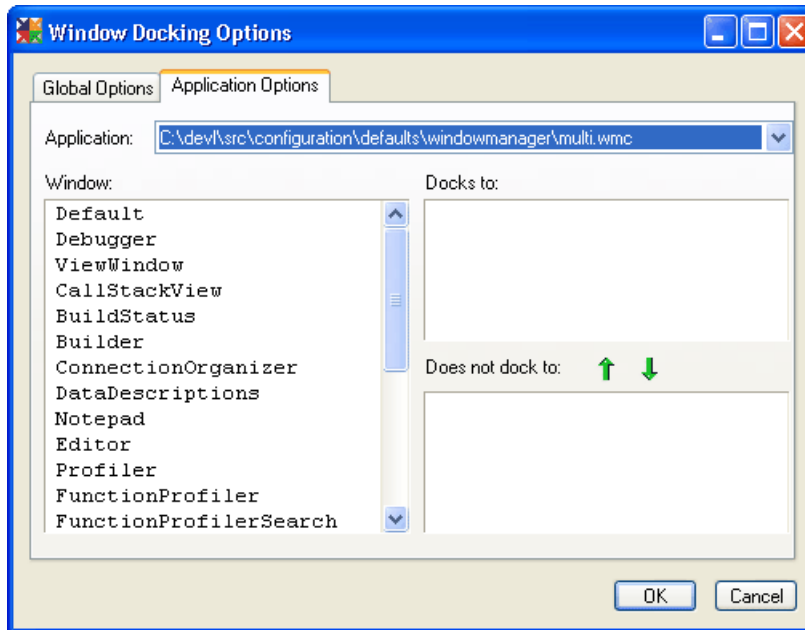
Compatibility

UNIX only

Disables window docking behavior that may not be supported in specific circumstances. Available options are:

- **Disable grouping in KDE** — Disables window grouping when the KDE window manager is detected. This setting overrides any application-level options and is selected by default. For more information, see “UNIX Docking Limitations” on page 200.
- **Disable activation follows focus** — Does not raise all windows in a group when one of the windows gets the focus. This option is selected by default.
- **Disable geometry caching** — Disables caching of window manager geometry. Selecting this option causes a small window to open and close each time an application is started, enabling MULTI to detect window manager geometry properties. This option is cleared by default and should not be selected unless MULTI cannot detect a change in window managers.

Window Docking Application Options



The items on the **Application Options** tab allow you to configure window grouping on a per-application basis.

Application

Displays the location of the window docking configuration file that is currently being viewed. This file may be located in the site-wide configuration directory or in the user configuration directory. Any changes are saved to the user configuration directory.

Window

Lists available window types (which vary depending on the **Application** setting). Each window supports configurable window grouping.

Docks to

Specifies the window types that form groups with **Window**.

You can use the arrow buttons to move window types between the **Docks to** and **Does not dock to** lists. If window type A is added to the **Docks to** list for window type B, then window type B should also be added to the **Docks to** list for window type A. If this is not done, window groups are only formed when you move window type B near window type A, but not vice versa.

Does not dock to

Specifies the window types that do not form groups with **Window**.

You can use the arrow buttons to move window types between the **Docks to** and **Does not dock to** lists.

Other General Configuration Options

The options in this section are not accessible from the **Options** window. To set these options, you can either enter them in a configuration file or in the Debugger command pane (preceded by the **configure** command). For information about setting these options in configuration files, see “Creating and Editing Configuration Files” on page 175. For information about setting configuration options from the command pane, see the introduction of Chapter 9, “Configuration Options”.

blinkingCursor [on | off]

UNIX only

Permitted settings for this option are:

- **on** — [default] The vertical bar cursor in various windows (such as the Debugger command pane and the Editor) will blink.
- **off** — The cursor will be a solid non-blinking bar.

clearKeys

Removes all of the key bindings so they can be created from scratch with **keybind**.

clearMenus

Removes all menus so they can be created from scratch using **menu**.

clearMice

Removes all mouse button bindings so they can be created from scratch with the **mouse** command. See “Customizing Mouse Behavior with the mouse Command” on page 193.

clickPause *time*

UNIX only

Specifies the length of time, in *time* tenths of a second, that MULTI waits between successive button presses to recognize double or triple clicks. For example, if *time* is 4, two clicks on the same button, in the same mouse location, within four tenths of a second, are treated as a double-click. If the two clicks are separated by more than *time* tenths of a second, they are treated as two single-clicks.

If no mouse binding requiring multiple clicks exists for a particular mouse location, MULTI executes the mouse binding immediately without waiting for multiple clicks.

The default *time* is 4.

configureFile *filename*

Used within a configuration file to read in another configuration file specified by *filename*. After *filename* is processed, processing of the original configuration file continues as normal.

disAsmStyle [remote | XORmacs | unix]

Controls the style of disassembled Motorola 68000 series code. Permitted settings for this option are:

- **remote** — [default] Use **XORmacs** style if the code is destined for execution on an embedded processor or **UNIX** style for native development.
- **XORmacs** — Always use **XORmacs** style (such as `MOVE.L (12,A6),D0`).
- **unix** — Always use **UNIX/Sun** style (such as `move1 a6@(12),d0`).

focusOnRaise [on | off]

UNIX only

Controls whether existing windows are automatically given focus when they are raised. This option does not affect windows when they are displayed for the first time.

- **on** — [default] Existing windows are automatically given focus when they are raised. This setting is recommended if your window manager uses a click-to-focus policy.
- **off** — Existing windows are not given focus when they are raised. This setting is recommended if your window manager uses a focus-follows-mouse policy.

grabTimeout *time*

UNIX only

Specifies an interval used to force processes to release any X-server keyboard or mouse grabs:

- If *time* is less than zero — **MULTI** does not check to see if there are any outstanding grabs on the X-server each time it stops.
- If *time* is greater than zero — **MULTI** checks to see if the process has any outstanding grabs when the process is stopped by a signal or breakpoint. If both the keyboard and the mouse are grabbed, **MULTI** waits *time* seconds before aborting the other process's grabs, and then continues debugging, allowing you to use the mouse and keyboard to issue commands to the Debugger.

The default *time* is -1.

This option is useful for Solaris native development only.

iconify [on | off]

UNIX only

Permitted settings for this option are:

- **on** — The next MULTI window will be minimized as an icon when it opens.
If you repeatedly configure this option to **on**, it will remember the number of times you have done this, and cause the next *num* windows to be created minimized (one per time you configure the option to **on**). After the correct number of windows come up minimized, this option resets itself to **off**.
- **off** — [default] Does not minimize MULTI windows.

This option is only applied to the next window created and does not affect existing windows.

ignoreMotion *num*

Specifies the number of pixels the mouse can move during a press and release of a mouse button.

If you move the mouse less than *num* pixels between a button press and release, the click is treated as a single click. If you move the mouse more than *num* pixels between press and release, the mouse click is no longer treated as a single click but as two independent press and release events.

The default *num* is 4.

keyBind

Used to assign an action to a key. For more information, see “Customizing Keys and Mouse Behavior” on page 191 and “Customizing Keys with the keybind Command” on page 191.

menu

Used to define a menu. For more information, see “Configuring and Customizing Menus” on page 186.

menuDelay *time*

The delay, in milliseconds, between moving the cursor over a submenu entry and the submenu opening. If set to 0, submenus will open immediately. The default *time* is 25.

mouse

Used to assign an action to mouse button clicks. For more information, see “Customizing Keys and Mouse Behavior” on page 191 and “Customizing Mouse Behavior with the mouse Command” on page 193.

multiiconPreName *string*

UNIX only

When you iconify a window, *string* prefixes the icon name. For example, if you set *string* to `MULTI`, iconified Editor windows are named **MULTI:Filename.c** instead of **Filename.c**.

If your window manager does not support iconifying, this option has no effect. Changes to this setting only affect new windows.

The default is an empty string.

multiWinPreName *string*

UNIX only

Prepends *string* to the title bar's window name. The default is an empty string.

Changes to this setting only affect new windows.

noDecoration [on | off]

UNIX only

Permitted settings for this option are:

- `on` — If the windows manager supports this setting, all windows appear without title bars.
- `off` — [default] Windows appear with title bars.

numberSeparator *string*

When printing, MULTI breaks up large numbers with underscores to ease reading. For example, `0x123456789` might print as `0x1_23456789`. This option allows you to specify an alternative character (*string*) to use in place of the underscore. *String* may only be a single character. Specifying no character (**NumberSeparator ""**) results in MULTI not using any separator character to break up numbers.

showGrepCommand [true | false]

Permitted settings for this option are:

- `true` — The full **grep** command prints out whenever the **grep** command is executed.
- `false` — [default] The full **grep** command does not print out.

synchronous [on | off]

UNIX only

Permitted settings for this option are:

- `on` — Enable synchronous X-windows mode. Synchronous mode ensures that X-window calls from MULTI complete before they return. This can be useful when running MULTI on faulty X-servers.
- `off` — [default] Disables synchronous X-windows mode.

useWmPositioning [on | off]

Permitted settings for this option are:

- `on` — MULTI allows the window manager to determine where all windows should go.
- `off` — [default] MULTI automatically places windows in convenient locations.

Project Manager Configuration Options

This section describes configuration options that affect the Project Manager. To access the following options, select **Config** → **Options** → **Project Manager** tab.

For information about setting configuration options from the command pane, see the introduction of Chapter 9, “Configuration Options”.

Open build details window

Permitted settings for this option are:

- **Always (showProgress always)** — The **Build Details** window opens every time you build an application, even if no warnings or errors are produced.
- **On Warnings and Errors (showProgress on warnings and errors)** — [default] The **Build Details** window opens if your build produces warnings or errors.
- **On Errors (showProgress on errors)** — The **Build Details** window opens if your build produces errors, but does not open for warnings.
- **Never (showProgress never)** — The **Build Details** window never opens, even if warnings or errors are produced.

Automatically open editor on errors

Permitted settings for this option are:

- **Selected (autoEditErrors on)** — [default] If the **Build Details** window is open or is configured to open on errors, the code that caused the first error of the build is automatically displayed in an Editor window.
- **Cleared (autoEditErrors off)** — Errors are displayed in the **Build Details** window. Double-click an error to open it in an Editor window.

All MULTI tools opened during the current MULTI session are notified of changes to this option. For more information, see “Propagating Configuration Settings” on page 174.

On warnings

Permitted settings for this option are:

- **Selected (autoEditWarnings on)** — If the **Build Details** window is open or is configured to open on warnings, the code that caused the first warning of the build is automatically displayed in an Editor window.
- **Cleared (autoEditWarnings off)** — [default] Warnings are displayed in the **Build Details** window. Double-click a warning to open it in an Editor window.

All MULTI tools opened during the current MULTI session are notified of changes to this option. For more information, see “Propagating Configuration Settings” on page 174.

Use MULTI Editor on errors (even if alternate editor is specified elsewhere)

Permitted settings for this option are:

- **Selected (alwaysUseMeToFixBuildErrors on)** — [default] Errors and warnings will be displayed in the MULTI Editor even if you have configured the MULTI environment to use an alternative, third-party editor.
- **Cleared (alwaysUseMeToFixBuildErrors off)** — Errors and warnings will be displayed in the editor you use in your MULTI environment.

All MULTI tools opened during the current MULTI session are notified of changes to this option. For more information, see “Propagating Configuration Settings” on page 174.

Project directory root

(defaultNpwDir *string*) Specifies the base directory where new projects are created. If left blank, new projects are created at:

- Windows — **C:\Documents and Settings\username\My Documents\My Projects**
- UNIX — **~/MyProjects/**

The project wizard creates a **Projectn** directory in this location to act as the root directory of the project.

All MULTI tools opened during the current MULTI session are notified of changes to this option. For more information, see “Propagating Configuration Settings” on page 174.

Parallel Build**Single-Thread Build**

Parallel Build (buildType parallel) — Builds programs using the number of processes specified in the **Number of parallel processes** text field.

Single-Thread Build (buildType singleThread) — [default] Runs the build in a single process.

Number of parallel processes

(**numberOfParallelProcesses** *num*) Specifies the number of processes that may be run in parallel. This option is only available with the **Parallel Build** option. The default *num* is 2.

Use lock files (-lockout)

Permitted settings for this option are:

- **Selected (useLockFiles on)** — Prevents multiple simultaneous builds of the same program or library by creating temporary **.lck** lock files.
- **Cleared (useLockFiles off)** — [default] Does not create temporary **.lck** lock files.

Note: **-lockout** is the corresponding **gbuild** option. For information about the **gbuild** utility program, see the *MULTI: Building Applications* book for your target processor family.

Execute tools at low priority (-nice)

Permitted settings for this option are:

- **Selected (useLowPriority on)** — Executes the build with lower than normal priority—scheduling priority 10 (UNIX) or the idle priority class (Windows).
- **Cleared (useLowPriority off)** — [default] Executes the build at normal priority.

Note: **-nice** is the corresponding **gbuild** option. For information about the **gbuild** utility program, see the *MULTI: Building Applications* book for your target processor family.

Show tool commands (-commands)

Permitted settings for this option are:

- **Selected (toolCommands on)** — Displays commands as they are executed.
- **Cleared (toolCommands off)** — [default].

Note: **-commands** is the corresponding **gbuild** option. For information about the **gbuild** utility program, see the *MULTI: Building Applications* book for your target processor family.

Automatically share target connections

Permitted settings for this option are:

- **Selected (autoConnectionMode on)** — [default] The primary target connection is used.
- **Cleared (autoConnectionMode off)** — Multiple target connections can run concurrently. Advanced users who need to have multiple target connections running must clear this option before starting multiple connections.

Display connection type in Connection Chooser

Permitted settings for this option are:

- **Selected (displayConnectionTypeinChooser on)** — The type of a connection method is displayed in brackets after the name in the drop-down list in the **Connection Chooser** dialog. Custom connection methods have no type displayed.
- **Cleared (displayConnectionTypeinChooser off)** — [default] Only the name of the connection method is displayed.

Debugger Configuration Options

This section describes configuration options that affect the Debugger. Most of these options appear on the **Debugger** tab of the **Options** window.

The Debugger Options Tab

To access the following options, select **Config** → **Options** → **Debugger** tab.

For information about setting configuration options from the command pane, see the introduction of Chapter 9, “Configuration Options”.

Ask before halting to set breakpoint

Permitted settings for this option are:

- **Selected (verifyHalt on)** — [default] The Debugger will ask for confirmation before halting the process to set a breakpoint.
- **Cleared (verifyHalt off)** — The Debugger will automatically halt, set the breakpoint, and continue the process without requiring user intervention.

Use procedure relative line number (vs. file relative)

Permitted settings for this option are:

- **Selected (procRelativeLines on)** — [default] Debugger commands, such as the **e** command, interpret line numbers as procedure relative, instead of file relative.
- **Cleared (procRelativeLines off)** — Line numbers are treated as file relative.

You can always specify file-relative line numbers regardless of the setting of this option by using the **#** character before line number arguments to commands.

Display all numbers/characters as hex

Permitted settings for this option are:

- **Selected (hexMode on)** — All numeric values are displayed in hexadecimal format.
- **Cleared (hexMode off)** — [default] Display format is based on the “natural” display format for that type. For integral types, the natural display format is decimal.

View unsigned char as integer

Permitted settings for this option are:

- **Selected (viewUnsignedCharAsint on)** — [default] The “natural” display format of **unsigned chars** will be the same as for **ints**. This is useful when you want to view byte values as numeric values instead of characters.
- **Cleared (viewUnsignedCharAsint off)** — The natural display format for unsigned chars is a literal character, such as “A”.

Coloring for multiple debuggers

Specifies how the background color of new Debugger windows is chosen. It is useful to turn this on when using multiple Debugger windows to make them visually distinct. Permitted settings for this option are:

- **Do Not Color (backgroundMode off)** — All Debugger windows use the normal background color.
- **Use Color Offsets (backgroundMode offset)** — [default] New Debugger windows will use predetermined offsets from the normal background color. This option is usually best since it will pick colors near the current background color, and keep the text as legible as possible.
- **Use Preset Colors (backgroundMode preset)** — New Debugger windows will use a color from a set of pre-chosen, highly distinctive colors.

Line numbers in source pane

Specifies how line numbers are displayed on the left side of the Debugger source pane. Permitted settings for this option are:

- **No Number (lineNumberMode none)** — No line numbers are displayed.
- **File Number (lineNumberMode file)** — The file relative line numbers are displayed.
- **Proc Number (lineNumberMode proc)** — The procedure relative line numbers are displayed.
- **Both Numbers (lineNumberMode both)** — [default] Both file-relative and procedure-relative line numbers are displayed.

Show variable values in tooltips

Permitted settings for this option are:

- **Never (fastest and safest) (hoverValues never)** — Never show variable values in tooltips.
- **On Mouse Hover (hoverValues always)** — When the mouse is positioned over a variable, show its value in a tooltip.
- **On Mouse Hover with Shift Key (hoverValues onShift)** — [default] When the mouse is positioned over a variable and the shift key is pressed, show its value in a tooltip.

Debugger child windows

Windows only

Controls how Debugger child windows (such as the Data Explorer and the **Breakpoint** window) interact with the Debugger window in terms of stacking order and focus. Permitted settings for this option are:

- **Stay above Debugger (viewsAreChildrenMode children)** — Child windows are always displayed in front of the Debugger, and the keyboard focus stays with the Debugger when a child window appears. This makes it easy to keep track of child windows and keeps the focus with the Debugger for further command input, but can be annoying if the child windows start to obscure the Debugger window. Child windows can be minimized, but this mode precludes partially obscuring a child window with the Debugger to save space.
- **Can be below Debugger (viewsAreChildrenMode topLevel)** — [default] Child windows can be either above or below the Debugger, and keyboard focus remains with the Debugger for further command input when a child window first appears. Because the keyboard focus remains with the Debugger, child windows will appear to flash above the Debugger and then immediately go behind it, so this option is most useful on a large screen where child windows do not overlap the Debugger when they first appear.
- **Are initially focused (viewsAreChildrenMode keepFocus)** — Child windows can be either above or below the Debugger, and also receive the keyboard focus when they first appear. This keeps child windows from immediately going behind the Debugger, but takes the focus away from the Debugger. Further command input into the Debugger is not possible until you click the Debugger again to give it the keyboard focus.

Command pane prompt

(**prompt string**) Specifies the string used as a prompt in the Debugger command pane to indicate that user input is desired. The default *string* is `MULTI>`.

Configure Debugger Buttons

Opens a dialog box that you can use to edit the Debugger toolbar buttons. This dialog box is only accessible from the MULTI Debugger. For more information, see “Adding, Removing, and Rearranging Toolbar Buttons” in Appendix A, “Debugger GUI Reference” in the *MULTI: Debugging* book.

More Debugger Options

This button opens the **More Debugger Options** dialog box, which contains additional configuration options for the Debugger (see “The More Debugger Options Dialog” on page 234).

Minimum initial size (WxH)

(**minViewSize widthxheight**) Specifies the minimum initial size of Data Explorers in characters (width) by lines (height).

Data Explorers are auto-sized. This option provides a lower limit that sets how small a Data Explorer can be, but not necessarily how small it is. The default *width x height* is 40x3.

Maximum initial size (WxH)

(**maxViewSize widthxheight**) Specifies the maximum initial size of Data Explorers in characters (width) by lines (height).

Data Explorers are auto-sized. This option provides an upper limit that sets how large a Data Explorer can be, but not necessarily how large it is. Even if the **minViewSize** option (preceding) is greater in either dimension than **maxViewSize**, the **maxViewSize** value is still used.

The default *width x height* is 40x40.

Initial position (XxY)

(**firstPosition widthxheight**) Specifies, in characters (width) by lines (height), the initial Data Explorer position from the top-left of the screen. To give the coordinates in pixels, put a p after them (for example, 100x100p).

This option only applies to the first Data Explorer. Subsequent Data Explorers are offset from the first one that was created, avoiding excessive overlap. If this option is left unspecified (blank), the Data Explorer remembers its previous position. The default is blank.

The More Debugger Options Dialog

To access this dialog box, select **Config** → **Options** → **Debugger** tab and click the **More Debugger Options** button.

For information about setting configuration options from the command pane, see the introduction of Chapter 9, “Configuration Options”.

Automatically dereference pointers

Permitted settings for this option are:

- **Selected (derefPointer on)** — [default] MULTI automatically follows pointers when it encounters them and prints both the pointer value and the object it points to.
- **Cleared (derefPointer off)** — MULTI only displays the value of the pointer.

Check syntax of breakpoints when they are set

Permitted settings for this option are:

- **Selected (bpSyntaxChecking on)** — [default] Commands associated with a breakpoint must pass syntax checking for the breakpoint to be set. Breakpoint commands that fail syntax checking cannot be set.
- **Cleared (bpSyntaxChecking off)** — A syntax error in a breakpoint command will not be detected by MULTI until the breakpoint is hit and MULTI tries to execute the associated commands.

Continue running script files on error

Permitted settings for this option are:

- **Selected (continuePlaybackFileOnError on)** — The Debugger will not stop if it encounters an error while running a script file. Instead, it will ignore the error and continue reading and performing commands out of that file.
- **Cleared (continuePlaybackFileOnError off)** — [default] The Debugger aborts running a script file if it encounters an error.

“s” (step) and “n” (next) are blocking by default

Permitted settings for this option are:

- **Selected (blockStep on)** — Step/next operations block subsequent commands until the step/next operations have finished.
- **Cleared (blockStep off)** — [default] Step/next operations allow subsequent commands to execute before the step/next operations have finished. This can cause scripts that use step/next commands to behave inconsistently.

You can make a step or next command blocking or non-blocking regardless of the setting of this option by appending an `n` (for non-blocking) or `b` (for blocking) to the command. See “Single-Stepping Commands” in Chapter 14, “Program Execution Command Reference” in the *MULTI: Debugging Command Reference* book.

See also the **blockRun** option in “Other Debugger Configuration Options” on page 241.

Show locations of variables for “print” command

Permitted settings for this option are:

- **Selected (showAddress on)** — The location (address in memory or register name) of a variable is printed when using the **print** command.
- **Cleared (showAddress off)** — [default] The location of a variable will not be printed; only the value is displayed.

Display typedef type instead of basic type

Permitted settings for this option are:

- **Selected (leaveTypeDef on)** — Data Explorers for structures will display the name of the type definition
- **Cleared (leaveTypeDef off)** — [default] Data Explorers for structures will display the actual self-contained type.

Show position in non-GUI (-nodisplay) mode

Permitted settings for this option are:

- **Selected (showPosinNoDisplayMode on)** — [default] When running in non-GUI mode, MULTI will print the line of source associated with the current line pointer, each time the current line pointer moves. This is the line that will be affected by the next Debugger command executed.
- **Cleared (showPosinNoDisplayMode off)** — The line of source code associated with the current line will not display.

Pressing Esc halts the target when connected

Permitted settings for this option are:

- **Selected (escHalts on)** — [default] MULTI attempts to halt the current process when you press Esc and the Debugger window is active.
- **Cleared (escHalts off)** — MULTI does not attempt to halt the current process when you press Esc.

Prompt for OSA package when package is not found

Permitted settings for this option are:

- **Selected (requestOsaPackage on)** — If MULTI determines that the program is a freeze-mode debugging project (see Chapter 24, “Freeze-Mode Debugging and OS-Awareness” in the *MULTI: Debugging* book) but cannot determine the Operating System Awareness (OSA) package, MULTI will prompt you for the package name. You can choose the package name so that MULTI can provide the corresponding debugging features, or select **Cancel** to continue without OSA.
- **Cleared (requestOsaPackage off)** — [default] MULTI will not prompt you for the OSA package name and will continue without OSA.

Delete dead tasks from group

Permitted settings for this option are:

- **Selected (deleteDeadTaskFromGroup on)** — The Task Manager will delete dead task fingerprints from the task group. For information about task groups and the meaning of dead tasks, see Chapter 23, “Run-Mode Debugging” in the *MULTI: Debugging* book.
- **Cleared (deleteDeadTaskFromGroup off)** — [default] The Task Manager will not delete dead task fingerprints from the task group.

Automatically verify ROM image is up-to-date

Permitted settings for this option are:

- **Selected (autoVerifyRomSections on)** — When an executable residing in the target’s ROM begins executing, the Debugger will automatically check whether the host and target versions of the executable match. If the executable has a checksum, even slight variations between the two versions will be detected. If the executable does not have a checksum, only more significant differences will be detected. For more information, see Chapter 21, “Working with ROM” in the *MULTI: Debugging* book.

To generate a checksum for an executable, use the **-checksum** option. See Chapter 6, “Builder and Driver Options” in the *MULTI: Building Applications* book.

- **Cleared (autoVerifyRomSections off)** — [default] The Debugger will not check the target’s version of the executable.

Translate DWARF debugging information (requires dwarf2dbo license)

Permitted settings for this option are:

- **Selected (autoDwarf2Dbo on)** — Enables automatic translation of DWARF debugging information when an executable built with a third-party compiler that generates DWARF information is loaded. This option is only meaningful if you have licensed **dwarf2dbo**; otherwise it has no effect.
- **Cleared (autoDwarf2Dbo off)** — [default] Disables automatic translation of DWARF debugging information.

For more information about translating DWARF debugging information, see “Using the Debugger with Third-Party Compilers” in Appendix D, “Using Third-Party Tools with the MULTI Debugger” in the *MULTI: Debugging* book.

Translate stabs debugging information (requires stabs2dbo license)

Permitted settings for this option are:

- **Selected (autoStabs2Dbo on)** — Enables automatic translation of Stabs debugging information when an executable built with a third-party compiler that generates Stabs information is loaded. This option is only meaningful if you have licensed **stabs2dbo**; otherwise it has no effect.
- **Cleared (autoStabs2Dbo off)** — [default] Disables automatic translation of Stabs debugging information.

For more information about translating Stabs debugging information, see “Using the Debugger with Third-Party Compilers” in Appendix D, “Using Third-Party Tools with the MULTI Debugger” in the *MULTI: Debugging* book.

Reuse Data Explorer

Permitted settings for this option are:

- **Selected (unifyViewWindows on)** — Causes MULTI to display newly opened variables in an existing Data Explorer when available.
- **Cleared (unifyViewWindows off)** — Causes each newly viewed variable to be displayed in a separate Data Explorer.

For more information, see Chapter 9, “Viewing and Modifying Variables with the Data Explorer” in the *MULTI: Debugging* book.

All MULTI tools opened during the current MULTI session are notified of changes to this option. For more information, see “Propagating Configuration Settings” on page 174.

Stepping over longjumps

When the Debugger performs a *next operation* over a subroutine that calls **longjmp** or throws a C++ exception (implemented by a call to **longjmp** internally), the subroutine does not return in the normal way because of the transfer of control within the function call. This is significant because the Debugger uses a temporary breakpoint just after the normal return address to implement the next operation. This is also true of step if there is no source available for the subroutine. When **longjmp** is called, this temporary breakpoint is bypassed because of the transfer of control, and execution can “run away” since the breakpoint will never be hit.

This configuration option controls how MULTI attempts to properly catch unexpected transfers of control while stepping through code. Permitted settings for this option are:

- **Ignore/Run Away (longjmpStepMode ignoreRunAway)** — [default] Allows the code to call **longjmp** without attempting to detect abnormal transfers of control.
- **Minimize Temp Stops (longjmpStepMode minimizeTempStops)** — Fixes the problem in a way that does not cause temporary stops in **longjmp** as the process runs normally. This option inserts and removes a temporary breakpoint at **longjmp** for each next over a function call.
- **Maximize Step Speed (longjmpStepMode maximizeStepSpeed)** — Fixes the problem in a way that minimizes the time it takes to do a next. This option leaves a permanent breakpoint at **longjmp** and results in execution always halting due to a breakpoint being hit whenever the process calls **longjmp**—even during normal execution.

Debug server timeout in seconds

(**serverTimeout seconds**) Specifies the number of seconds to wait for communication from a debug server before assuming it is dead and disconnecting from it.

A timeout that is too low may cause a premature disconnect from the debug server, and is not recommended. A fairly high timeout can be useful for very slow debug servers or for debug servers that are being debugged. However, a high timeout can be frustrating if the debug server dies because the Debugger cannot accept input while it’s waiting for communication from the debug server.

The default is 15 seconds.

Interval to refresh Task Manager (in seconds)

(**serverPollinterval seconds**) Specifies the polling interval in seconds. MULTI checks for interesting events happening in the underlying RTOS and refreshes the **Task Manager** periodically.

The default is 1 second for INTEGRITY and 5 for other real-time operating systems.

Criteria to decide if a task is in a group

Controls how to match a task fingerprint against a task. For more information about how the fingerprint is used in the construction process of a task group, see Chapter 23, “Run-Mode Debugging” in the *MULTI: Debugging* book.

Permitted settings for this option are:

- **Task Name (taskMatchCriteria name)** — [default] Matches against the name of the task.
- **Task Identifier (taskMatchCriteria id)** — Matches against the identifier of the task.
- **Task Name or Identifier (taskMatchCriteria nameOrId)** — Matches against the name or identifier of the task.

Prompt when exiting Debugger

Controls whether a confirmation dialog box appears when a user attempts to close the Debugger. Permitted settings for this option are:

- **Confirm If Process Started (promptQuitDebugger normal)** — [default] The confirmation dialog box is displayed only if the Debugger is attached to a process that has been started. If the Debugger is attached to a process that has not been started, the Debugger closes without displaying the confirmation dialog box. In a run-mode debugging environment, the Debugger closes without displaying the confirmation dialog even though the process has been started.
- **Always Confirm (promptQuitDebugger always)** — The Debugger will not close until the user confirms the action in the confirmation dialog box.
- **Never Confirm (promptQuitDebugger never)** — The Debugger never displays the confirmation dialog box before closing.

Prompt when program changes

Controls whether a dialog box appears when the Debugger detects that the program being debugged has changed. Permitted settings for this option are:

- **Always Prompt (promptProgramChange prompt)** — [default] A dialog box is displayed when the Debugger detects that the program being debugged has changed. The dialog allows the program to be reloaded or for the change to be ignored. If ignored, some debugging actions, such as viewing another source file, may cause out of date warnings to be printed to the command pane.
- **Reload Without Prompting (promptProgramChange reload)** — The Debugger will reload the program being debugged when a change is detected. See the **debug**.
- **Ignore Program Changes (promptProgramChange ignore)** — The Debugger ignores any program changes. Some debugging actions, such as viewing another source, may cause out of date warnings to be printed to the command pane.

Maximum size for container display

(**maxContainerDisplaySize** *num*) Sets the maximum number of elements initially displayed when a container is viewed in a Data Explorer. The default *num* is 20.

Increment to maximum container size

(**containerSizeincrement** *num*) Sets the number of elements to add when expanding a container that is being viewed in a Data Explorer. The default *num* is 10.

Other Debugger Configuration Options

The options in this section are not accessible from the **Options** window. To set these options, you can either enter them in a configuration file or in the Debugger command pane (preceded by the **configure** command). For information about setting these options in configuration files, see “Creating and Editing Configuration Files” on page 175. For information about setting configuration options from the command pane, see the introduction of Chapter 9, “Configuration Options”.

allowExecutioninBpCommand [on | off]

- **on** — Stepping and execution of command line procedure calls are allowed from within a breakpoint command. This is somewhat risky because infinite breakpoint command recursion can occur if the execution from within the breakpoint command causes the same or another breakpoint to be hit. The **resume** command is always allowed in a breakpoint command.
- **off** — [default].

allowProcCallinOsaTask [on | off]

- **on** — Permits command line procedure calls via a freeze-mode connection to an operating system kernel in any context. This may lead to overwritten register values and should only be used if known to be safe.
- **off** — [default] Disallows command line procedure calls via a freeze-mode connection to an operating system kernel if MULTI detects an executing task. This restriction prevents MULTI from overwriting register values of non-current tasks when attempting to restore the register file after a procedure call. This could happen if a context switch occurred during the procedure call and the kernel did not completely restore the register file when switching back to the task where you initiated the call (as is common practice with floating-point registers on tasks that do not use floating point).

attemptToShowOldVersionOfUpdateSource [on | off]

- **on** — The Debugger will display the versions of source files that were used to build the executable being debugged by using the version control history to check out the appropriate version of the file.
- **off** — [default].

blockRun [on | off]

- `on` — Run/continue operations block subsequent commands until the run/continue operations have finished.
- `off` — [default] Run/continue operations allow subsequent commands to execute before the run/continue operations have finished. This can cause scripts that use run/continue commands to behave inconsistently.

See “Run Commands” and “Continue Commands” in Chapter 14, “Program Execution Command Reference” in the *MULTI: Debugging Command Reference* book.

See also the **blockStep** option in “The More Debugger Options Dialog” on page 234.

clearButtons

Removes all the Debugger buttons (except for the **Close Debugger** button if it is present) so that you can create them from scratch with the **debugbutton** command (see “Configuring and Customizing Toolbar Buttons” on page 183) or the **Customize Toolbar** window (see “Adding, Removing, and Rearranging Toolbar Buttons” in Appendix A, “Debugger GUI Reference” in the *MULTI: Debugging* book).

cTextSize *num*

Sets the maximum number of scroll back lines available in the **Cmd**, **Trg**, **I/O**, **Srl**, **Py**, and **Tfc** panes.

If a MULTI session is in use for a long period of time, these panes can use a large amount of memory. Setting this option to a smaller value reduces memory usage, but also limits the number of available scroll back lines.

The valid range for *num* is 1024 – 10485760. The default *num* is 524288.

debugButton

Lists the defined Debugger buttons. See also “Configuring and Customizing Toolbar Buttons” on page 183.

downloadWindow [true | false]

The download window shows the current progress of a program download. Permitted settings for this option are:

- `true` — [default] The download window will be used when downloading a program to the debug server.
- `false` — The download window will not be used.

echoCommandsFromPlaybackFiles [true | false]

- `true` — When executing from a playback file, Debugger commands being executed will be displayed.
- `false` — [default] When executing from a playback file, Debugger commands being executed will not be displayed.

formatStringMaxDepth *num*

Limits the depth to which nested structs are displayed in a Data Explorer. The valid range for *num* is 1 – unlimited (0x7fffffff). The default *num* is 0x7fffffff.

formatStringMaxLength *num*

Limits the length of the value displayed in each row of a Data Explorer. For example, if this is set to 2000, only the first 2000 characters of a string will display, followed by . . . to indicate there are more characters that are not displayed. Data values displayed in a single row in a Data Explorer can be very long (when displaying a long string value or a structure with many nested structs, for example).

The valid range for *num* is 1024 to 10 megabytes (10485760). The default *num* is 8192.

geometry *widthxheight* [[+*x_offset*+*y_offset*] | [-*x_offset*-*y_offset*]]

Sets the size and position of the Debugger window, where:

- *width* and *height* — Correspond to the pixel size of the Debugger window.
- *x_offset* and *y_offset* — Are optional values that indicate the pixel offset at which the Debugger window appears from the top-left corner of the screen for the + variant or from the bottom-right corner of the screen for the - variant. For example, the entry: **geometry 500x700+0+0** specifies a 500-pixel-wide by 700-pixel-high Debugger window that appears in the top-left corner of the screen.

The defaults depend on screen size and vary from system to system. The *width* is approximately wide enough to display 80 characters on a line.

globalHeading [on | off]

- on — Enables the global scope entry in the Browse window.
- off — [default] Disables global scope entry in the Browse window.

All MULTI tools opened during the current MULTI session are notified of changes to this option. For more information, see “Propagating Configuration Settings” on page 174.

gotoHitsBpAtTargetAddress [on | off]

- on — If you are using the Debugger command **g**, any breakpoint at the destination will be hit as soon as execution begins at the new location. See the **g** command in “General Program Execution Commands” in Chapter 14, “Program Execution Command Reference” in the *MULTI: Debugging Command Reference* book.

This option also controls whether a breakpoint set on the first instruction of a procedure called from the command line is hit.

- off — [default] The breakpoint will not be hit.

history *num*

This option specifies how many commands to remember for the Debugger command history mechanism. For more information, see “History Commands” in Chapter 15, “Scripting Command Reference” in the *MULTI: Debugging Command Reference* book.

The default *num* is 256.

iconGeometry *widthxheight+x_offset+y_offset*

UNIX only

Specifies the dimensions (ignored by some window managers) for the icon that is used when the Debugger is minimized, where:

- *width* and *height* — Are the width and height of the icon in pixels.
- *x_offset* and *y_offset* — Specify the number of pixels that the icon should be offset from the top-left corner of the screen.

The defaults are 32x64+0+0.

implicitEvalEcho [on | off]

Permitted settings for this option are:

- *on* — [default] The value of an expression is echoed when the expression is entered into the command pane.
- *off* — The value of an expression is not echoed when the expression is entered into the command pane.

Note: This option affects expression evaluation in the `%EVAL{commands}` sequence used with the **substitute** command. For more information, see the **substitute** command in “Command Manipulation and Macro Commands” in Chapter 15, “Scripting Command Reference” in the *MULTI: Debugging Command Reference* book.

interleavedOutput [true | false]

Controls whether output from Debugger panes other than the command pane (**Cmd**) will also be output to the command pane. Permitted settings for this option are:

- *true* — [default] Output from Debugger panes other than the command pane will be output to the command pane. For example, **I/O** pane output will show up in the command pane with the prefix **I/O:**.
- *false* — Output from other panes will not appear in the command pane.

linesNonOverlapped *num*

By default, when the Debugger opens a Data Explorer or monitor window with **useWmPositioning off** (see “Other General Configuration Options” on page 221), the new window is stacked on top of previous windows to save screen space. This option leaves the top *num* lines of the previous window visible. The default *num* is 4.

osaSwitchToUserTaskAutomatically [true | false]

Controls whether the Debugger automatically displays the currently executing user-mode task when the kernel is stopped while executing user-mode tasks. (This is in addition to automatically displaying all of the tasks in the system as controlled by the **osaTaskAutoAttachLimit** configuration option below.)

Permitted settings for this option are:

- `true` — [default] Automatically displays the currently executing user-mode task.
- `false` — Does not display the currently executing user-mode task.

See also “Debugging in Freeze Mode” in Chapter 24, “Freeze-Mode Debugging and OS-Awareness” in the *MULTI: Debugging* book.

osaTaskAutoAttachLimit *num*

Specifies the maximum number of OSA tasks to display in the Debugger target list.

If the tasks have been loaded from trace data and the number of tasks exceeds this limit, one task per AddressSpace is displayed in the target list. For information about manually displaying additional tasks loaded from trace, see “Saving and Loading a Trace Session” in Chapter 17, “Analyzing Trace Data with the TimeMachine Tool Suite” in the *MULTI: Debugging* book.

If the tasks are not loaded from trace, they are displayed consecutively until this limit is reached. For example, if *num* is set to 32 and there are 33 tasks, the 33rd task is not displayed in the target list. For information about manually displaying additional tasks that have not been loaded from trace, see “Debugging in Freeze Mode” in Chapter 24, “Freeze-Mode Debugging and OS-Awareness” in the *MULTI: Debugging* book.

The default *num* is 32.

paddedHex [true | false]

Permitted settings for this option are:

- `true` — Forces hex values to be displayed padded to their full width with zeros. For example, a four-byte variable with a value of 0x8 is displayed as 0x00000008.
- `false` — [default] Hex values are displayed as entered.

prepareAllCores [on | off]

Permitted settings for this option are:

- **on** — When you prepare a single core of a multi-core target (for example, via the **prepare_target** command), MULTI automatically prepares each remaining core as if you had run **prepare_target -verify=none** on the executable associated with it. (This is equivalent to choosing the **Prepare Target** dialog option **Program already present on target. Verify: Not at all** for each remaining core.) For more information, see “Preparing Multiple Cores to Run a Single Executable” in Chapter 24, “Freeze-Mode Debugging and OS-Awareness” in the *MULTI: Debugging* book.

See also the **-allcores** option to the **prepare_target** command in “General Target Connection Commands” in Chapter 18, “Target Connection Command Reference” in the *MULTI: Debugging Command Reference* book.

- **off** — [default] Each core of a multi-core target is treated independently when you prepare the target.

See also the **-onecore** option to the **prepare_target** command.

procQualifiedLocalimpliesOutermostBlock [on | off]

When the current line pointer is in an inner block of a procedure and that inner block defines a variable that has the same name as a variable in the outer block, a procedure qualified reference to the name of the variable can either refer to the variable of that name in the outermost block or to the variable of that name in the current inner block. This option determines which of the two variables is used.

Permitted settings for this option are:

- **on** — The variable in the outermost block is used.
- **off** — [default] The variable in the current inner block is used.

quietTogCmd [on | off]

Permitted settings for this option are:

- **on** — The **tog** command does not echo the status of the breakpoint(s) it toggles.
See also the **q** option for the **tog** command in Chapter 4, “Breakpoint Command Reference” in the *MULTI: Debugging Command Reference* book.
- **off** — [default].

recordCommentedCommandsFromMacros [true | false]

When command recording is enabled, commands executed from playback files are recorded as comments. This option controls whether commands executed from macros are also recorded as comments. Permitted settings for this option are:

- **true** — Commands executed from macros are recorded as comments.
- **false** — [default] Commands executed from macros are not recorded as comments.

runRcScripts [on | off]

Permitted settings for this option are:

- **on** — [default] Runs **.rc** scripts when a new program is debugged.
- **off** — Does not run **.rc** scripts when a new program is debugged. See also the **-norc** command line option in Appendix C, “Command Line Reference” in the *MULTI: Debugging* book.

setBpAtAdrinitWhenExecing [on | off]

- **on** — [default] MULTI sets a breakpoint at the starting routine of a program when the process is run. Execution is resumed automatically when this breakpoint is hit unless a user breakpoint was set at the same spot. This is necessary to prevent corrupted call stack traces.
- **off** — MULTI does not set a breakpoint at the starting routine of a program when the process is run.

sharedSymbols [true | false]

- **true** — [default] MULTI attempts to debug shared objects.
- **false** — MULTI does not attempt to debug shared objects.

shellConfirm [on | off]

Permitted settings for this option are:

- **on** — [default] If MULTI is running in the background, the **shell** command opens a dialog box that allows you to confirm or cancel the execution of the shell command before the shell command executes. This is helpful when MULTI is running in the background because the shell commands have no accessible standard output or standard error.
- **off** — MULTI always executes shell commands immediately, without opening the dialog box.

See also the **shell** command in Chapter 15, “Scripting Command Reference” in the *MULTI: Debugging Command Reference* book.

silentlyReloadSymbols [on | off]

Permitted settings for this option are:

- **on** — The Debugger automatically kills any existing process and reloads the symbols without displaying a dialog box. This is useful if you are using a Debugger primarily to examine symbols and not to debug running processes.
- **off** — [default] When the executable open in the Debugger changes, the Debugger displays a dialog box asking if any existing process should be killed and the symbols reloaded.

stepToBpignoresResumeinBpCmd [true | false]

Controls whether the **c** (continue) command in a breakpoint is ignored. For more information, see the **c** command in “Continue Commands” in Chapter 14, “Program Execution Command Reference” in the *MULTI: Debugging Command Reference* book.

Note: This option determines whether a continue command is ignored, not a resume command.

Permitted settings for this option are:

- **true** — MULTI ignores the **c** command in the breakpoint command list, prints a warning, and stops the process at the breakpoint.
- **false** — [default] MULTI does not ignore the **c** command in the breakpoint command list. For example, if you have a breakpoint with the command list **print x; c** and you step on to this breakpoint, the process starts running because the **c** command causes the process to continue. The **c** command inside a breakpoint always causes the process to start running, which is incorrect behavior if you want to step on to that line.

Note that the **resume** command should be used inside breakpoint command lists. If the breakpoint’s command list is **print x; resume**, the process stops when you step on to the breakpoint. If the process runs and hits the breakpoint, it will keep running (the process resumes whatever it was going to do before hitting the breakpoint). For more information, see the **resume** command in Chapter 4, “Breakpoint Command Reference” in the *MULTI: Debugging Command Reference* book.

targetWindowSwitchViewOnBpHit [on | off]

Controls whether the Debugger automatically switches to a task that is not currently selected in the target list when a breakpoint is hit in that task. Permitted settings for this option are:

- **on** — [default] When a breakpoint is hit in a task that is not currently selected in the target list, the Debugger automatically switches to that task if no other Debugger window is open on it and if the currently selected task is not halted. Switching occurs for both software and hardware breakpoints.
- **off** — The Debugger does not automatically switch to a task when a breakpoint is hit in that task.

tbTypeBg *color*

tbTypeFg *color*

Specifies the background color (commands end with **Bg**) or foreground color (commands end with **Fg**) for Tree Browser displays, where:

- *Type* is one of the following:
 - **FunctionNormal** (**tbFunctionNormalBg** and **tbFunctionNormalFg**) — Functions displayed in the static call Tree Browser which have debugging information.
 - **FunctionNoinfo** (**tbFunctionNoinfoBg** and **tbFunctionNoinfoFg**) — Functions displayed in the static call Tree Browser which do not have debugging information.
 - **FunctionRecursive** (**tbFunctionRecursiveBg** and **tbFunctionRecursiveFg**) — Functions displayed in the static call Tree Browser that may be recursive.
 - **FunctionAdrTaken** (**tbFunctionAdrTakenBg** and **tbFunctionAdrTakenFg**) — Nodes representing the possibility of calls to a function through function pointer in a Tree Browser.
 - **DynNormal** (**tbDynNormalBg** and **tbDynNormalFg**) — Functions displayed in the dynamic call Tree Browser with debugging information.
 - **DynNoinfo** (**tbDynNoinfoBg** and **tbDynNoinfoFg**) — Functions displayed in the dynamic call Tree Browser without debugging information.
 - **FileNormal** (**tbFileNormalBg** and **tbFileNormalFg**) — Files displayed in the file call Tree Browser with debugging information.
 - **FileNoinfo** (**tbFileNoinfoBg** and **tbFileNoinfoFg**) — Files displayed in the file call Tree Browser without debugging information.
 - **ClassUnion** (**tbClassUnionBg** and **tbClassUnionFg**) — Unions displayed in the class Tree Browser.
 - **ClassStruct** (**tbClassStructBg** and **tbClassStructFg**) — Types displayed in the class Tree Browser with debugging information.
 - **ClassNoinfo** (**tbClassNoinfoBg** and **tbClassNoinfoFg**) — Types displayed in the class Tree Browser without debugging information.
 - **ClassClass** (**tbClassClassBg** and **tbClassClassFg**) — Class data types displayed in the class Tree Browser.
 - **ClassEnum** (**tbClassEnumBg** and **tbClassEnumFg**) — Enum data types displayed in the class Tree Browser.
- **Bg** or **Fg** — Specifies whether you are changing the background or foreground color of the displayed items.
- *color* — Specifies, in RGB (**## ## ##**) format, the new color.

viewDef [*number_option*] [*eval_option*] [[-]*other_options*]...

Specifies how data will be displayed in Data Explorers, where:

- *number_option* can be one of the following:
 - `naturalOrHex` — Numbers display in their default states unless **hexMode on** is set (see “The Debugger Options Tab” on page 230).
 - `natural` — Numbers display in their default states.
 - `hex, dec, oct, binary` — Numbers and characters display in base 16, base 10, base 8, or base 2 notation, respectively.
- *eval_option* specifies how the Debugger re-evaluates expressions when updating Data Explorers, and can be one of the following:
 - `reEvaluate` — Each expression is re-evaluated in the same context as when the Data Explorer displaying it was first created.
 - `reEvalContext` — Expressions are re-evaluated within the current procedure at the top of the call stack.
 - `reEvalinGlobal` — When expressions are re-evaluated, only looks for variables in the global scope.
 - `useAddress` — Uses last valid address of variables being displayed.
- *more_options* can include any of the options listed below. (Prepending an option with a dash (-) disables the behavior.)
 - `showAddress` — Displays the address (rather than name) of the variable.
 - `showFType` — Shows the type of each variable, class, or structure.
 - `alternate` — Displays data in an alternative format, if available.
 - `padHex` — Adds zeros to the left of hexadecimal numbers to maintain same bit width, as needed. (Otherwise, only non-zero hexadecimal digits display.)
 - `showBases` — Displays the base classes of a C++ class along with a derived instance.
 - `showAllFields` — Displays all the fields of a complicated structure.
 - `showDerived` — Determines the most derived C++ class type of the current object and redisplay the object cast to that type.
 - `expandValue` — Displays the value in memory (if in readable memory) that a pointer references.
 - `openPointer` — Dereferences all pointers.
 - `showChanges` — Highlights values that have changed.

The defaults are `naturalOrHex reEvalContext -showAddress -showFType -alternate -padHex showBases -showAllFields showDerived expandValue openPointer showChanges`.

warnOnBpReplacement [on | off]

Permitted settings for this option are:

- `on` — The Debugger displays a warning before replacing a pre-existing breakpoint with a new breakpoint. This can help prevent the accidental loss of breakpoints with long command lists.
- `off` — [default] The Debugger does not warn you before replacing a pre-existing breakpoint.

warnOnCmdAdrLinePromotion [on | off]

Permitted settings for this option are:

- `on` — The Debugger displays a warning when setting a breakpoint on a line with no corresponding assembly code.
- `off` — [default].

MULTI Editor Configuration Options

This section describes configuration options that affect the appearance and behavior of the MULTI Editor. (If you are using an alternative editor, see “Third-Party Editor Configuration Options” on page 213.) Most of these options appear on the **MULTI Editor** tab of the **Options** window.

The MULTI Editor Options Tab

To access the following options, select **Config** → **Options** → **MULTI Editor** tab.

For information about setting configuration options from the command pane, see the introduction of Chapter 9, “Configuration Options”.

Reuse editor windows

Permitted settings for this option are:

- **Selected (`openFilesinNewBuffers on`)** — The first file opened from a MULTI executable such as the Project Manager or Debugger opens in a new Editor window. Subsequently opened files reuse the Editor window initially opened from the MULTI executable.
- **Cleared (`openFilesinNewBuffers off`)** — [default] Every file opens in a new Editor window.

All MULTI tools opened during the current MULTI session are notified of changes to this option. For more information, see “Propagating Configuration Settings” on page 174.

Create backup files when saving

Permitted settings for this option are:

- **Selected (`editorBackups on`)** — A backup of the on-disk version of a file will automatically be created before saving over it. The backup file has the same name as the original file, with a tilde (~) appended to it.
- **Cleared (`editorBackups off`)** — [default] Backup files will not be created.

Allow middle click to paste

Permitted settings for this option are:

- **Selected (`allowMiddleClick on`)** — [default for UNIX] Clicking the middle mouse button will have the same effect as pasting from the selection buffer.
- **Cleared (`allowMiddleClick off`)** — [default for Windows] Clicking the middle mouse button will not paste from the clipboard.

Enter spaces in place of tabs

Permitted settings for this option are:

- **Selected (`tabsAreSpaces on`)** — Tab characters entered into an Editor window will be replaced by an appropriate number of space characters in the Editor buffer, regardless of whether the Tab is the result of pressing Tab, entering a **paste** command, or auto indenting in the Editor.
- **Cleared (`tabsAreSpaces off`)** — [default] Space characters are not inserted in place of Tab characters.

Drag and drop text editing

Permitted settings for this option are:

- **Selected (`dragAndDrop on`)** — Text can be moved in the Editor by selecting a block of text and dragging the mouse to a new location.
- **Cleared (`dragAndDrop off`)** — [default] Text cannot be moved by dragging the mouse.

Allow overtype mode

Permitted settings for this option are:

- **Selected (allowOvertyping on)** — Allows switching between insert and overtype mode.
- **Cleared (allowOvertyping off)** — [default] Does not allow switching between insert and overtype mode.

Tab size

(**tabSize spaces**) Specifies the number of spaces used to display a Tab in the Editor. The default is 8 spaces.

Indent size

(**editindent spaces**) Specifies the number of spaces in an indentation. The default is 4 spaces.

Ctrl+cursor jump size

(**editSomeSize num**) Specifies the value used by the **UpSome**, **DownSome**, **LeftSome**, and **RightSome** Editor commands. For information about these commands, see “Navigation Commands” on page 296.

The default *num* is 5.

Configure Editor Buttons

Opens the **Configure Buttons** dialog box, which allows you to edit the Editor buttons using the same format as the **editbutton** command (see “Configuring and Customizing Toolbar Buttons” on page 183). The dialog box lists currently defined buttons on the left, and available icons on the right.

More Editor Options

Opens the **More Editor Options** dialog box, which contains additional configuration options for the Editor. See “The More Editor Options Dialog Box” on page 257.

Auto indent as you type

Permitted settings for this option are:

- **Selected (aiimplicitindent on)** — [default] The Editor automatically indents the file as you type.
- **Cleared (aiimplicitindent off)** — The Editor will not automatically indent your file.

Only auto indent line if first char typed is # or *

Controls auto indentation. This configuration option only takes effect if **Auto indent as you type** (preceding) is selected.

Permitted settings for this option are:

- **Selected (aiimplicitOnlyAtinitial on)** — [default] The MULTI Editor only automatically indents a line when the first character (includes space characters) typed on the line is # or *.
- **Cleared (aiimplicitOnlyAtinitial off)** — The MULTI Editor automatically makes indenting adjustments when you type auto-indent characters, regardless of where they appear on the line. See “Auto-Indent Characters” on page 37.

Auto indent comments as you type

Controls auto indentation in comments. This configuration option only takes effect if **Auto indent as you type** (preceding) is selected.

Permitted settings for this option are:

- **Selected (aiimplicitindentinComments on)** — [default] Comments will be indented when auto indenting multiple lines.
- **Cleared (aiimplicitindentinComments off)** — Comments are not modified unless you auto indent a single line.

Double indent size when indenting body of switch

Permitted settings for this option are:

- **Selected (aiSwitchinTwo on)** — [default] For C or C++ source files, switch bodies are indented two levels so that case labels are indented one level in from the switch. For Ada source files, select bodies are indented two levels so that labels are indented one level in from the select.
- **Cleared (aiSwitchinTwo off)** — For C or C++ source files, the case labels are even with the switch. For Ada source files, the labels are even with the select.

Indent comments when indenting multiple lines

Permitted settings for this option are:

- **Selected (aiTouchComments on)** — Comments are indented when auto indenting multiple lines.
- **Cleared (aiTouchComments off)** — [default] Comments are only indented if you auto indent a single line.

Comments stick flush left

Permitted settings for this option are:

- **Selected (aiCommentsStayFlushLeft on)** — [default] All comments will be kept directly on the left margin.

To indent a comment that is stuck to the left margin while this option is selected, insert a space just before the start of the comment, then auto indent the comment.

To move a comment to the left margin, regardless of the position where the comment started, insert a number sign (#) as the first character of a comment. For example, if you are coding in C, type: /*# and the comment is automatically moved to the left margin.

- **Cleared (aiCommentsStayFlushLeft off)** — Indents will be applied to comments.

C chars aligned like '*' in comments

(aiCharsLikeStarinComment *char_string*) Allows characters other than asterisks (*) to line up in comments as if they were asterisks. The default *char_string* is - (dash).

The default setting allows correct automatic indentation of comments that have a column of - (dashes) down the left side, lined up under the * in the /* sequence. To have characters other than asterisks (*) line up in comments as if they were asterisks, make them part of the *char_string*.

C paren indent mode

Controls how the Editor indents a line in a source file if it starts within an open parenthesis/close parenthesis pair. Permitted settings for this option are:

- **Even with parentheses (aiParenindentMode evenWithParen)** — If there is a non-whitespace character between the open parenthesis and the end of its line, the lines enclosed in parentheses start at the same column as that character. Otherwise, the lines enclosed in parentheses start in the column just after the open parenthesis. For example:

```
void foo (void)
{
    bar(
        argument_one,
        argument_two
    );
}
```

- **Indent in two (aiParenindentMode indentinTwo)** — [default] Lines enclosed in parentheses start two indent levels in from the line that contains the open parenthesis. For example:

```
void foo (void)
{
    bar(
        argument_one,
        argument_two
    );
}
```

Ada paren mode

Controls how the Editor indents a line in an Ada source file if it starts within an open parenthesis/close parenthesis pair. Permitted settings for this option are:

- **Even with parentheses (aiAdaParenindentMode evenWithParen)** — [default] If there is a non-whitespace character between the open parenthesis and the end of its line, the lines enclosed in parentheses start at the same column as that character. Otherwise, the lines enclosed in parentheses start in the column just after the open parenthesis.
- **Indent in two (aiAdaParenindentMode indentinTwo)** — The lines enclosed in parentheses start two indent levels in from line that contains the open parenthesis.

The More Editor Options Dialog Box

To access this dialog box, select **Config** → **Options** → **MULTI Editor** tab and click the **More Editor Options** button.

For information about setting configuration options from the command pane, see the introduction of Chapter 9, “Configuration Options”.

Print 2 columns if landscape orientation is selected

Windows only

Permitted settings for this option are:

- **Selected (editPrint2Column on)** — The Editor prints files with two columns per sheet if landscape orientation is selected.
- **Cleared (editPrint2Column off)** — [default] The Editor prints one column per sheet.

Temp file directory

(tempFileDir *dirpath*) Specifies the directory where MULTI stores its temporary files. The default is blank, which has the following results:

- **Windows** — Uses the values of the `TMP` or `TEMP` environment variables and defaults to the current directory if those variables are not set.
- **UNIX** — Uses the values of the `TMPDIR` or `TEMP` environment variables and defaults to `/tmp` if those variables are not set.

Initial width in characters

(editWidth *width*) Specifies the initial width of Editor windows in characters.

This option is only useful when **Save window positions and sizes** is set to **Cleared (rememberWindowPositions off)**. See the **Save window positions and sizes** option in “The General Options Tab” on page 208.

The default *width* is 80.

Initial height in characters

(editHeight *height*) Specifies the initial height of Editor windows in characters.

This option is only useful when **Save window positions and sizes** is set to **Cleared (rememberWindowPositions off)**. See the **Save window positions and sizes** option in “The General Options Tab” on page 208.

The default *height* is 32.

Selection margin width in pixels

(**selectionMarginWidth** *num*) Specifies the width of the left margin in the Editor in pixels. If the width is 0, the left margin does not appear and entire lines of text can no longer be selected by using the margin. Changing the selection margin width only affects new Editor windows. The default *num* is 13.

Generate auto-recover file every ... seconds

(**editincrFrequency** *num*) Specifies the number of seconds between generation of auto-recover files.

The Editor will create an auto-recover file every *num* seconds as you edit. If the power goes out or the Editor crashes, the next time you open the same file, you will be given the option to re-apply the edits saved in the auto-recover file.

The default *num* is 120.

Per File Settings Defaults

The settings in this section are used when a file is first opened. Changes to the default settings will automatically be applied to all files open in the Editor. To change the settings for an individual file without affecting other open files, choose **View** → **Per File Settings** in the Editor.

Spaces per indent for Ada

(**adaindentSize** *num*) Specifies the number of spaces in an indentation for Ada files. The default *num* is 3.

Ada continuation line indent

(**adaContinuationSize** *num*) Specifies the number of spaces in an indentation for a continuation line in Ada files. The default *num* is 2.

Word wrap

Permitted settings for this option are:

- **Selected (wordWrap on)** — If lines are longer than the wrapping width, the Editor will automatically split lines on word boundaries as you type.
- **Cleared (wordWrap off)** — [default] Lines will not wrap.

Wrap column

(**wrapColumn** *width*) When **Word wrap** is selected (**wordWrap on**), *width* specifies the last column a character can occupy before the Editor wraps to the next line.

The default *width* is 79.

Wrap indent offset

(**wrapindent num**) When a word wraps to the next line, the word is automatically indented *num* extra spaces from where it would normally appear. The default *num* is 2.

Fill Paragraph Column

(**fillParagraphColumn num**) Specifies the column at which each line will be wrapped when using Ctrl + Shift + A to reformat comments.

The default *num* is 80.

Other MULTI Editor Options

The options in this section are not accessible from the **Options** window. To set these options, you can either enter them in a configuration file or in the Debugger command pane (preceded by the **configure** command). For information about setting these options in configuration files, see “Creating and Editing Configuration Files” on page 175. For information about setting configuration options from the command pane, see the introduction of Chapter 9, “Configuration Options”.

autoGrabHeadFiles [on | off]

Controls whether the MULTI Editor automatically grabs function prototypes from a source file’s **include** files when the source file is loaded.

- **on** — [default] The MULTI Editor automatically grabs function prototypes when opening source files. This setting enables auto-completion of function names.
- **off** — The MULTI Editor does not grab function prototypes when opening source files. This setting is recommended if you notice poor performance when this option is enabled.

clearEditButtons

Removes all Editor buttons so they can be created from scratch with the **editbutton** command (see “Configuring and Customizing Toolbar Buttons” on page 183).

drawWrapLine [on | off | asWordWrap]

Specifies whether or not to draw the wrap line in the MULTI Editor. Permitted settings for this option are:

- **on** — [default] Enables the wrap line.
- **off** — Disables the wrap line.
- **asWordWrap** — Enables the wrap line if word wrap is enabled.

This option does the same thing as the Editor command **DrawWrapLine**.

editButton

Creates a new Editor button. This option does the same thing as the Debugger command **editbutton**. For usage and information, see “Configuring and Customizing Toolbar Buttons” on page 183.

editParenMatch *time*

Every time you type a right parenthesis, right square bracket, or right curly brace, the Editor briefly selects the matching one. The Editor will pause on this highlight of the previous match for *time* tenths of a second. A value of 0 (zero) will disable matching. The default *time* is 10.

Session Configuration Options

This section describes configuration options that allow you to control what information from the current session MULTI remembers for use in future sessions. To access the following options, select **Config → Options → Session** tab.

For information about setting configuration options from the command pane, see the introduction of Chapter 9, “Configuration Options”.

Save command history between sessions

Permitted settings for this option are:

- **Selected (saveCommandHistory on)** — [default] The command history for each executable will be saved. If the executable has not been debugged before, it will have no previous command history.
- **Cleared (saveCommandHistory off)** — The command history will not be saved.

Save data explorers between sessions

Permitted settings for this option are:

- **Selected (saveViewWindows on)** — Saves all variables being viewed in Data Explorers between MULTI sessions on the same executable. When restored, the variables are all added to one Data Explorer, or each is added to a different Data Explorer as dictated by the **Reuse Data Explorer** option. (See the description of this option in “The More Debugger Options Dialog” on page 234.)
- **Cleared (saveViewWindows off)** — [default] Data Explorer content is not saved.

Save arguments between sessions

Permitted settings for this option are:

- **Selected (saveRunArguments on)** — [default] Saves the last arguments supplied to run the target program between MULTI sessions on the same executable. To run the executable with no arguments, use the **R** command. See the **R** command in “Run Commands” in Chapter 14, “Program Execution Command Reference” in the *MULTI: Debugging Command Reference* book.
- **Cleared (saveRunArguments off)** — Arguments are not saved.

Save debugger window position

Permitted settings for this option are:

- **Selected (saveDebuggerWindowPos on)** — Saves and restores the size and position of the MULTI Debugger window on a per executable basis. Note that the Debugger window is only sized and positioned on window creation, so debugging a different executable from an already opened MULTI will not have any effect.
- **Cleared (saveDebuggerWindowPos off)** — [default] Debugger window position will not be saved.

Remember breakpoints

Determines whether the Debugger remembers software breakpoints (including shared object breakpoints) that you have set for a program the next time you debug the same program. Permitted settings for this option are:

- **Never (rememberBreakpoints never)** — The Debugger clears all software breakpoints whenever you load or reload a program.
- **Across Reloads (rememberBreakpoints withinSession)** — [default] The Debugger remembers software breakpoints when you reload the program within a single session. Software breakpoints are lost when you exit MULTI.
- **Across Sessions (rememberBreakpoints acrossSessions)** — The Debugger remembers software breakpoints even if you exit and restart MULTI.

Note: The Debugger may not remember group breakpoints and does not remember hardware breakpoints.

Restored breakpoints overwrite breakpoints set by scripts

Permitted settings for this option are:

- **Selected (overwriteScriptBreakpoints on)** — Previously saved breakpoints will overwrite breakpoints that have been created by a script run automatically for the current executable.
- **Cleared (overwriteScriptBreakpoints off)** — [default] Previously saved breakpoints will not overwrite breakpoints that have been created by a script run automatically for the current executable.

Remember base addresses (e.g. _TEXT)

Determines whether the Debugger remembers address offsets that you have set for a program the next time you debug the same program. This option is only applicable to programs that are compiled for use with position-independent code and/or position-independent data. Permitted settings for this option are:

- **Never (rememberBaseAddr never)** — The Debugger will clear all base addresses whenever you load or reload a program.
- **Across Reloads (rememberBaseAddr acrossReload)** — [default] The Debugger will remember address offsets that you have set when you reload the program within a single session. Address offsets are lost when you exit MULTI.
- **Across Sessions (rememberBaseAddr acrossSessions)** — The Debugger will remember address offsets that you have set for a program even if you exit and restart MULTI.

You can set address offsets by assigning built-in variables `_DATA` and `_TEXT` to the Debugger or by using the command line options **-data offset** and **-text offset**. For more information about system variables, see “System Variables” in Chapter 12, “Using Expressions, Variables, and Procedure Calls” in the *MULTI: Debugging* book. For more information about command line options, see Appendix C, “Command Line Reference” in the *MULTI: Debugging* book.

Remember last connect command used for process

Permitted settings for this option are:

- **Selected (saveDebugServer on)** — [default] Saves the most recently used Connection Method on a per-executable basis. If the executable is started and MULTI is not connected, the saved Connection Method will be used as the default in the **Connection Chooser** dialog box.
- **Cleared (saveDebugServer off)** — The last Connection Method is not saved.

Automatically save changed connection files

Permitted settings for this option are:

- **Selected (autoSaveConnectionsInFiles on)** — [default] MULTI saves any outstanding changes to all Target Connection files loaded in the Connection Organizer when you exit.
- **Cleared (autoSaveConnectionsInFiles off)** — Changes to Target Connection files will not be automatically saved.

Automatically save “User Methods” changes

Permitted settings for this option are:

- **Selected (autoSaveUserConnections on)** — [default] MULTI saves any outstanding changes to the “User Methods” Target Connection file when you exit.
- **Cleared (autoSaveUserConnections off)** — All changes to the “User Methods” Target Connection file will be lost, even if **Automatically save changed connection files** is **Selected (autoSaveConnectionsInFiles on)**.

Directory Memory

Determine which directory to initially display in the **File Chooser**. Permitted settings for this option are:

- **Across Sessions (rememberDirs rememberAcross)** — **File Chooser** will always open in the previously viewed directory for the same operation, even after closing and restarting MULTI. The user directories will only be used on the very first run of MULTI, or if a user settings file cannot be found.
- **Within Sessions (rememberDirs rememberWithin)** — [default] **File Chooser** will open in the appropriate previously viewed directory, but it will not remember directories viewed in previous sessions. **File Chooser** will use the User directories initially each time MULTI is restarted.
- **None (rememberDirs rememberNever)** — **File Chooser** will always open with the appropriate User directory, and never remember which directory was previously viewed.

User Directories

Opens the **User Directories** dialog box, which sets the directories the file chooser starts with if directory memory is not available or is turned off.

To change any of the settings in this dialog box, select the check box, then enter the full path of the directory.

General Files:

- **Selected** (**genFileSet true**) — Opens the **General Files** directory field, where you can enter the path.
- **Cleared** (**genFileSet false**) — [default] The *current working directory* is used.
- **General Files** directory (**genFilesDir *pathname***) — [default is *current working directory*] Use this field to enter the directory path for general files.

Source Files:

- **Selected** (**sourceFileSet true**) — Opens the **Source Files** directory field, where you can enter the path.
- **Cleared** (**sourceFileSet false**) — [default] The *current working directory* is used.
- **Source Files** directory (**sourceFilesDir *pathname***) — [default is *current working directory*] Use this field to enter the directory path for source files.

Project Files:

- **Selected** (**buildFileSet true**) — Opens the **Project Files** directory field, where you can enter the path.
- **Cleared** (**buildFileSet false**) — [default] The *current working directory* is used.
- **Project Files** directory (**buildFilesDir *pathname***) — [default is *current working directory*] Use this field to enter the directory path for project files.

Executables/Binaries:

- **Selected** (**execFileSet true**) — Opens the **Executables/Binaries** directory field, where you can enter the path.
- **Cleared** (**execFileSet false**) — [default] The *current working directory* is used.
- **Executables/Binaries** directory (**execFilesDir *pathname***) — [default is *current working directory*] Use this field to enter the directory path for executables.

Debug Servers:

- **Selected** (**debugServerSet true**) — Opens the **Debug Servers** directory field, where you can enter the path.
- **Cleared** (**debugServerSet false**) — [default] The *current working directory* is used.
- **Debug Servers** directory (**debugServersDir *pathname***) — [default is *current working directory*] Use this field to enter the directory path for debug servers.

Show version control information on file chooser dialog box

UNIX only

Permitted settings for this option are:

- **Selected (showVersionControl true)** — If a user has the file checked out, the **File Chooser** dialog box will display the name of that user next to the filename in a column titled **Version Control**.
- **Cleared (showVersionControl false)** — The **Version Control** column will be hidden and the version control system will not be queried.

Warning: Selecting this option causes MULTI to query the version control system for each file. On some version control systems, turning on this option might slow down **File Chooser** performance significantly.

Colors Configuration Options

This section describes configuration options that you can use to change the display colors for MULTI. To access the following options, select **Config** → **Options** → **Colors** tab.

To edit colors, double-click the colored box next to any option. The **Color Chooser** opens (for more information, see “The Color Chooser” on page 269).

On Windows, color options are expressed in RGB format as three decimal values. For example, 255 0 0 specifies the color red. In the tables that follow, the default RGB values provided are in hexadecimal, but decimal values appear on the **Colors** tab for Windows users.

On UNIX, color options are expressed in RGB hexadecimal format as a number sign (#) followed by the six-digit hexadecimal value. For example, #FF0000 specifies the color red.

Unless otherwise stated, all MULTI tools opened during the current MULTI session are notified of changes to color options. For more information, see “Propagating Configuration Settings” on page 174.

For information about setting configuration options from the command pane, see the introduction of Chapter 9, “Configuration Options”.

Load Color Scheme

UNIX only

Opens the **Color Scheme** dialog box, which you can use to select from a list of predefined color schemes. If you prefer a different appearance from the default color scheme, you can try these to find the one that is closest to what you want.

Global Colors

UNIX only

Global colors can only be set on UNIX systems. They affect the default colors for user interface elements. The following global color options may be set:

- **Background (background color)** — [default is #FFFFFF] Background color of “user areas,” such as the color behind the text you enter.
- **Foreground (foreground color)** — [default is #000000] Color of text.
- **Control Area (controlColor color)** — [default is #C0C0C0] Background color of “control areas,” such as menu bars and buttons.
- **Selection (Select color)** — [default is #000080] Background color of text selections. The foreground color of selections is based on the **foreground** setting and is chosen automatically.

Debugger Colors

Controls how certain elements are displayed in the Debugger. The colors you can customize are:

- **Assembly (assembly color)** — [default is #0000FF] Color of interlaced assembly code.
- **Break Dot (bDotColor color)** — [default is #00CD00] Color of break dots, which indicate locations where breakpoints may be set.
- **Status (breakColor color)** — [default is #FF0000] Color of the process status and debug server messages (“STOPPED,” “RUNNING,” etc.).
- **Context Arrow (pointerColor color)** — [default is #0000FF] Color of the context arrow, which indicates what context to use for commands in the Debugger.
- **File line #**
 - **Selected (useFileRelLineBg true)** — [default] Enables file relative line numbers.
 - **Cleared (useFileRelLineBg false)** — Disables file relative line numbers.
 - **Color field (fileRelLineBg color)** — [default is #FFFFFF] Color of the background behind the column of file relative line numbers in the source pane.
- **Proc line #**
 - **Selected (useProcRelLineBg true)** — [default] Enables procedure relative line numbers.
 - **Cleared (useProcRelLineBg false)** — Disables procedure relative line numbers.
 - **Color field (ProcRelLineBg color)** — [default is #e4e4e4] Color of the background behind the column of procedure relative line numbers in the source pane.

Syntax Coloring

Permitted settings for this option are:

- **Selected (colorSyntax on)** — [default] Source code will be syntax colored based upon the **Syntax Color Settings** listed below.
- **Cleared (colorSyntax off)** — Source code will not be syntax colored.

If you change the setting of this option but do not save the configuration as the default, only the MULTI tool from which the option was changed is aware of the new setting. If you do save the configuration as the default, however, all MULTI tools launched after the change are also notified of the new setting.

Syntax Color Settings

Colors may be set for the following items:

- **Comments** (**comment color**) — [default is #008000]
- **Keywords** (**keyword color**) — [default is #0000FF]
- **Dead Code** (**deadCode color**) — [default is #808080] Color of code enclosed in an `#if 0 ... #endif` block.
- **Numbers** (**number color**) — [default is #E000E0]
- **Strings** (**string color**) — [default is #800000]
- **Characters** (**character color**) — [default is #C000C0]
- **Customized** (**customized color**) — [default is #008080]

Color C++ comments in C

Permitted settings for this option are:

- **Selected** (**cppCommentsinC on**) — [default] C++ style comments (//) will be syntax colored as comments in both C++ and C source files.
- **Cleared** (**cppCommentsinC off**) — C++ style comments will not be syntax colored as comments in C source files.

If you change the setting of this option but do not save the configuration as the default, only the MULTI tool from which the option was changed is aware of the new setting. If you do save the configuration as the default, however, all MULTI tools launched after the change are also notified of the new setting.

More Color Options

Opens the **More Color Options** dialog box, which contains additional colors that you can configure. For more information, see “The More Color Options Dialog Box” on page 269.

The More Color Options Dialog Box

To access this dialog box, select **Config** → **Options** → **Colors** tab and click **More Color Options**.

For information about setting configuration options from the command pane, see the introduction of Chapter 9, “Configuration Options”.

Connection Organizer Colors
Sets the foreground color of the Connected Targets list, which is located in the Connection Organizer .
• Connected targets (startedConnectionFg color) — [default is #008000]
Diffview Colors
Sets the background color of the current selection in the Diff Viewer foreground.
• Highlight (diffHighlight color) — [default is #e0ffd0]

The Color Chooser

You can use the **Color Chooser** to select and customize colors for any of MULTI’s color options. To access the **Color Chooser**, double click any of the color boxes in the **Colors** tab of the **Options** window.

On Windows, the standard color chooser is used. For more information, see your Windows documentation.

On UNIX, the **Color Chooser** consists of a rectangle indicating the current color; three sliders to control the levels of red, green, and blue in the current color; and a number of preset basic colors.

1. To select a basic color — Click one of the colored boxes.

To select a custom color — Use the color sliders or the text fields beneath them to create a color.

2. Click **OK** to replace the color or **Cancel** to discard your changes and close the **Color Chooser**.

Deprecated Configuration Options

The following configuration options are deprecated and will not be supported in a future release. In MULTI 5, they are ignored.

- **allowExecutioninAssertionCommand**
- **assertionSyntaxChecking**
- **builderPosition**
- **debugButtonsLoc**
- **dontUseVc**
- **iconifiedButtons** (previously included in the **Options** window as **Use icons for buttons**)
- **maxWindowSize**
- **minWindowSize**
- **scrollBarWidth** (previously included in the **Options** window as **Scroll bar width in pixels**)
- **scrollHLocation** (previously included in the **Options** window as **Horizontal scroll bar**)
- **scrollLocation** (previously included in the **Options** window as **Vertical scroll bar**)
- **traceAlwaysOn**

Appendices

Part IV

Editor Commands

Appendix A

This Appendix Contains:

- Auto-Completion Commands
- Block Commands
- Clipboard Commands
- Configuration Commands
- Drag-and-Drop Commands
- File Commands
- Help Commands
- if Conditional Commands
- Indentation Commands
- Insert Commands
- Mode Commands
- Navigation Commands
- Search Commands
- Selection Commands
- Tag Commands
- Text Deletion Commands
- Tools Commands
- Undo/Redo Commands
- Version Control Commands
- View Commands
- Deprecated Commands

Most of the commands listed in this chapter are bound to Editor GUI menu items, keyboard shortcuts, and/or mouse buttons. This command reference is provided so you can customize the Editor to help you work more efficiently.

- If you do not want to use the mouse when you are editing, you can bind any command to a keystroke combination.
- If you find that you are often performing a particular sequence of actions, you can combine the commands for these actions into a single keystroke, key sequence, mouse button combination, GUI button, or menu item.

For more information about binding commands to menu choices, keystrokes, mouse clicks, and buttons, see Chapter 8, “Configuring and Customizing MULTI”. For a list of all default key and mouse bindings, see Appendix C.



Note You can also issue these commands manually by selecting **Tools** → **Execute Editor Commands** and entering them in the text box. However, since most of these commands are bound to menu choices, keystrokes, mouse clicks, and/or buttons, there should be little need to enter commands in this way.



Note Editor commands are case-insensitive, thus `DiffFiles` and `difffiles` are equivalent and are both valid. In this chapter, we use mixed-case notation where it makes the commands more readable.

Auto-Completion Commands

The MULTI Editor provides the following auto-completion commands. These commands function even when auto-completion is disabled.

AcceptAcString

Accepts the string auto-completed by the MULTI Editor.

The Editor deletes the auto-completed string if you click the mouse, type a character (depending on your key-stroke configuration), etc. before accepting the string.

By default, **AcceptAcString; MoveToEndOfSelectionAndUnselect** is bound to:

- Tab

AutoCompleteList

Displays a list of auto-completion strings from which you can choose an entry to complete the text at the cursor or, if only one match is possible, automatically completes the text at the cursor.

The maximum number of strings listed is determined by the following entry in the corresponding language definition file:

```
max_match = num
```

where *num* is the maximum number of matches to display. The default is 10.

By default, this command is bound to:

- Ctrl + /

AutoCompletePrototypeList

Displays a list of function prototypes matching the name of the function at the cursor. You can select an entry to complete the text at the cursor.

The maximum number of function prototypes listed is determined by the option **max_match** in the corresponding **.gsc** file.

By default, this command is bound to:

- Ctrl + '

DelUnconfirmedAcString

Deletes the string that has been auto-completed by the MULTI Editor but not accepted.

MoveToEndOfSelectionAndUnselect

Moves the cursor to the end of the selected text and clears the selection. There is no effect if there is no selection.

By default, **AcceptAcString; MoveToEndOfSelectionAndUnselect** is bound to:

- Tab

NextAutoCompleteString

Auto-completes the characters with the next string that matches the pattern.

By default, this command is bound to:

- Ctrl +]

PrevAutoCompleteString

Auto-completes the characters with the previous string that matches the pattern.

By default, this command is bound to:

- Ctrl + [

Block Commands

The following commands allow you to modify text in the current primary selection. Although they are designed to operate on selected text, most of these commands have a default behavior that occurs even when there is no selection.

CommentBlock

Inserts language-specific characters to signify that the selected text is a comment and not code. For more information, see “Working with Comments” on page 38.

By default, this command is bound to:

- **Comment** menu item (see “The Block Menu” on page 66)
- **Ctrl + ***

JoinLines

Joins two lines of text by replacing the new line character at the end of the current line and all initial whitespace on the next line with a single space.

By default, this command is bound to:

- **Join Lines** menu item (see “The Block Menu” on page 66)
- **Ctrl + P**

LowerCaseBlock

Changes all characters in the current selection to lowercase.

By default, this command is bound to:

- **LowerCase** menu item (see “The Block Menu” on page 66)
- **Ctrl + -** (**Ctrl + minus**)

UnCommentBlock

Removes comment-style characters from the selected text to make it active code. The selected text must begin with comment-start symbols and end with comment-stop symbols. For more information, see “Working with Comments” on page 38.

By default, this command is bound to:

- **UnComment** menu item (see “The Block Menu” on page 66)
- Ctrl + Shift + U

UpperCaseBlock

Capitalizes all characters in the current selection.

By default, this command is bound to:

- **UpperCase** menu item (see “The Block Menu” on page 66)
- Ctrl ++ (Ctrl + plus)

Clipboard Commands

MULTI provides four internal clipboards.

On Windows, the first clipboard is the same as the Windows clipboard. Text or data stored there is available to other Windows applications.

On UNIX, the content of the clipboard can be made available to other applications with the clipboard manager. **Launch clipboard manager** is a configurable option on the **Config** → **Options** → **General** tab (see “The General Options Tab” on page 208).

Copy n

Copies the selected text to clipboard number n . When this command is issued, the specified clipboard’s previous content is lost; other clipboards are unaffected.

By default, **Copy1** is bound to:

- **Copy** menu item (see “The Edit Menu” on page 59)
- Ctrl + C
- Copy or L6 (UNIX only)

By default, **Copy2** is bound to:

- Ctrl + Shift + C
- Shift + Copy or Shift + L6 (UNIX only)

On UNIX, **Copy3** is bound to:

- Meta + Ctrl + C
- Ctrl + Copy or Ctrl + L6

On UNIX, **Copy4** is bound to:

- Meta + Ctrl + Shift + C
- Ctrl + Shift + Copy or Ctrl + Shift + L6

Cut n

Deletes the selected text and places it on clipboard number n . When this command is issued, the specified clipboard's previous content is lost; other clipboards are unaffected.

By default, **Cut1** is bound to:

- **Cut** menu item (see "The Edit Menu" on page 59)
- Ctrl + X
- Cut or L10 (UNIX only)

Cut2 is bound to:

- Ctrl + Shift + X
- Shift + Cut or Shift + L10 (UNIX only)

On UNIX, **Cut3** is bound to:

- Meta + Ctrl + X
- Ctrl + Cut or Ctrl + L10

On UNIX, **Cut4** is bound to:

- Meta + Ctrl + Shift + X
- Ctrl + Shift + Cut or Ctrl + Shift + L10

NoSelection; ContinueSelection; Left; Cut2; Left; Paste2; Right

Transposes the previous two characters.

By default, this command is bound to:

- Ctrl + Shift + T

Paste n

Inserts text from clipboard number n .

By default, **Paste1** is bound to:

- **Paste** menu item (see “The Edit Menu” on page 59)
- Ctrl + V
- Paste or L8 (UNIX only)

Paste2 is bound to:

- Ctrl + Shift + V
- Shift + Paste or Shift + L8 (UNIX only)

On UNIX, **Paste3** is bound to:

- Meta + Ctrl + V
- Ctrl + Paste or Ctrl + L8

On UNIX, **Paste4** is bound to:

- Meta + Ctrl + Shift + V
- Ctrl + Shift + Paste or Ctrl + Shift + L8

RectCopy1

Copies a rectangular subsection of the current selection to clipboard number 1.

Only this clipboard is available for rectangular selections. For more information, see “Working with Columns” on page 39.

By default, this command is bound to:

- **Rect Copy** menu item (see “The Block Menu” on page 66)

RectCut1

Deletes a rectangular subsection of the current selection, and copies it to clipboard number 1. Only this clipboard is available for rectangular selections. For more information, see “Working with Columns” on page 39.

By default, this command is bound to:

- **Rect Cut** menu item (see “The Block Menu” on page 66)

RectPaste1

Pastes a clipboard selection created with **RectCopy1** or **RectCut1** to the current location. Only one clipboard is available for rectangular selections. For more information, see “Working with Columns” on page 39.

By default, this command is bound to:

- **Rect Paste** menu item (see “The Block Menu” on page 66)
- Ctrl + V

Configuration Commands

The following commands allow you to configure the Editor's appearance and behavior.

ClearConfig

Prompts before clearing the default configuration file.

By default, this command is bound to:

- **Clear User Default Configuration** menu item (see "The Config Menu" on page 73).

ConfigOptions

Opens the **Options** window.

By default, this command is bound to:

- **Options** menu item (see "The Config Menu" on page 73)

configure *config_option* [= | :] *value*

configure *config_option*

configure ?

Changes the value of a MULTI configuration option. The `configure ?` command displays a list of configurable options. You can separate *config_option* from *value* with an equal sign (=), a colon (:), or a whitespace character (). If *value* is a Boolean, you can omit it and MULTI will toggle the option's setting. For more information, see "Using the configure Command" on page 171.

CustomizeMenus

Opens the **Customize Menus** window, which allows you to configure menus in a number of MULTI tools. For information about how to use this window, see "Configuring and Customizing Menus" on page 186.

By default, this command is bound to:

- **Customize Menus** menu item (see "The Config Menu" on page 73)

LoadConfigFromFile

Opens the **Load Configuration from what file?** dialog box.

By default, this command is bound to:

- **Load Configuration** menu item (see "The Config Menu" on page 73)

SaveConfig

Saves the current configuration options in a file in the default location.

By default, this command is bound to:

- **Save Configuration as User Default** menu item (see “The Config Menu” on page 73).

SaveConfigToFile

Opens the **Save Configuration to what file?** dialog box.

By default, this command is bound to:

- **Save Configuration As** menu item (see “The Config Menu” on page 73)

Drag-and-Drop Commands

The Editor supports drag-and-drop actions such as moving text in an open file and, in Windows, dragging a file from a Windows Explorer to an Editor to open it. The drag-and-drop feature is controlled by the configuration option **dragAndDrop**, which is `off` by default (for more information, see “The MULTI Editor Options Tab” on page 251). The following commands perform various functions associated with this drag-and-drop behavior. Since they are all dependent upon the location of the mouse pointer, these commands are generally useful only when bound to mouse buttons and actions (see “Default Mouse Bindings” on page 336).

SelectionDrop

Completes the drag-and-drop or drag-and-drop-add operation. Legal drag spots are anywhere in the Editor text pane, except on the current selection, and are indicated by the mouse cursor changing into the **drop** (or **drop-add**) cursor. Illegal drop spots are indicated by the **no mouse** cursor.

Note: This command will be executed whenever the left mouse button is released during a drag-and-drop or a drag-and-drop-add operation.

SelectionStartDrag

Starts a drag-and-drop operation using the text of the primary selection. When the mouse moves over a legal drop spot, the cursor will change into the **drop** cursor, so you can drop the text to the new location. At the completion of the drag-and-drop operation, the text of the primary selection will be deleted from its original location, and pasted to the drop location. If there is no primary selection, this command has no effect.

Note: This command must be followed by a **SelectionDrop** command.

By default, this command is bound to:

- Left-click

SelectionStartDragAdd

Starts a drag-and-drop-add operation using the text of the primary selection. When the mouse is over a legal drop spot, the cursor will change into the **drop-add** cursor, so you can copy the text to a new location. At the completion of the drag-and-drop-add operation, the text will be copied to the drop location. If there is no primary selection, this command has no effect.

To execute this command, left-click and drag selected text, while holding down the Ctrl key. If you press Ctrl any time during a normal drag-and-drop operation, the operation will turn into a drag-and-drop-add operation.

Note: This command must be followed by a **SelectionDrop** command.

By default, this command is bound to:

- Ctrl + Left-click

File Commands

The following commands allow you to open, save, close, and otherwise manipulate files.

Close

Closes the current Editor window. If changes were made to open files, you will be prompted to save the files before closing.

By default, this command is bound to:

- **Close Editor** menu item (see “The File Menu” on page 56)
- Ctrl + Q

Done

Closes and saves changes to all open files without prompting, then exits the Editor.

By default, this command is bound to:

- **Save and Close** Toolbar button (see “The File Menu” on page 56).
- Ctrl + Shift + Q

DosFormat [-dos | -unix]

Sets the file format to DOS format or UNIX format. (The DOS format saves the file with carriage returns.) If you do not specify any arguments, this command toggles the file format.

By default, this command is bound to:

- **Edit → DOS Format** (see “The Edit Menu” on page 59).

EditorFlags

Opens the **Per File Settings** dialog box, which lists control settings for the current file. See “The Per File Settings Dialog Box” on page 64.

By default, this command is bound to:

- **Per File Settings** menu item (see “The View Menu” on page 62)

FileProperties

Displays the language name and other information the Editor is able to obtain about the currently edited file.

By default, this command is bound to:

- **Properties** menu item (see “The File Menu” on page 56)

LanguageOptions

Opens the **Language Settings** dialog box, which lists settings for the selected language.

By default, this command is bound to:

- **Per Language Settings** menu item (see “The View Menu” on page 62).

LoadFile [filename]

Opens the given file in either a new Editor window or the current Editor window, depending on the current setting of the **openFilesinNewBuffers** option (see “The MULTI Editor Options Tab” on page 251). If no parameters are specified, this command opens the **Edit File** dialog box for you to open or create a file.

LoadFileWithNewEditor [filename]

Opens the given file in a new Editor window. If no parameters are specified, this command opens the **Edit File** dialog box for you to open or create a file.

By default, this command is bound to:

- **New Editor** menu item (see “The File Menu” on page 56)
- Ctrl + N
- Ctrl + Shift + O

OpenFile [filename]

Opens the given file in the current Editor window. If no parameters are specified, this command opens the **Edit File** dialog box for you to open or create a file.

By default, this command is bound to:

- **Open** menu item (see “The File Menu” on page 56)
- Ctrl + O
- Open or L7 (UNIX only)

PageSetup

(Windows only)

Opens the **Page Setup** dialog box.

By default, this command is bound to:

- **Page Setup** menu item (see “The File Menu” on page 56)

Print

Opens the **Print** dialog box for you to print the file or current selection. For information about the UNIX **Print** dialog box, see “The Print Setup Dialog Box” on page 77.

By default, this command is bound to:

- **Print** menu item (see “The File Menu” on page 56)

QuerySaveAll

Opens the **SaveFiles?** dialog box, which lists all the currently edited files under version control. To choose to save any combination of these files, select the box next to the file's name. To save all of the selected files and type the same comment for all of them, click **OK**. This is equivalent to saving all of the files at once and entering the same comment for each log. All of the files also have exactly the same date and time in their log entry.

By default, this command is bound to:

- **Save All** menu item (see “The File Menu” on page 56)

QuerySaveComments

Identical to **QuerySaveAll**, except that if comments are turned off for a file under version control, it still prompts you for a comment.

Quit

Prompts you to save the changes made to all open files, then quits all currently running MULTI tools (including Debugger windows).

By default, this command is bound to:

- **Exit All** menu item (see “The File Menu” on page 56)

Revert

Reverts the file to the last saved version, discarding any changes that have not been saved.

By default, this command is bound to:

- **Revert to Saved** menu item (see “The File Menu” on page 56)

Save

Saves the current file.

By default, this command is bound to:

- **Save** menu item (see “The File Menu” on page 56)
- Ctrl + S

SaveAll

Automatically saves all open files without prompting.

SaveAs

Opens the **Save As** dialog box.

By default, this command is bound to:

- **Save As** menu item (see “The File Menu” on page 56)
- Ctrl + Shift + S

SelectLanguage [*language_index*]

Selects the language to use for color syntax, auto indentation, etc.

language_index is the sequential index (starting from 1) for each language defined in **index.gsc** (see “The index.gsc Configuration File” on page 47).

Note: The C and C++ syntax highlighting module attempts to gray out any code that is enclosed by the `#if 0` preprocessor directive. However, if several such directives are nested, only the outermost one will be highlighted correctly.

By default, this command is bound to:

- **Language** menu item (See “The View Menu” on page 62.)

ToggleReadOnly

Toggles the window permission for the current file between read-only and read/write mode, if possible. This does not affect the file’s permission on disk.

By default, this command is bound to:

- **Read Only Window** menu item (See “The View Menu” on page 62.)

ToggleWritePermission

Toggles the permission for the file between read-only and read/write mode, if possible. A stop sign in the right corner of the Editor’s status bar indicates that the file is currently read-only. This command changes both the file’s window permission and the file’s permission on disk.

By default, this command is bound to:

- **Toggle Write Permission** menu item (see “The File Menu” on page 56)

Help Commands

The following commands access the MULTI **Editor's** help features.

About

Opens the About dialog box.

By default, this command is bound to:

- | |
|---|
| <ul style="list-style-type: none">• About MULTI menu item (see “The Help Menu” on page 75) |
|---|

BugReport

Launches the gbugrpt utility program, which collects information about your MULTI installation and allows you to append it to a bug report form that you can fill out and email to Green Hills Technical Support.
--

Help

Opens MULTI's online help.

By default, this command is bound to:

- | |
|--|
| <ul style="list-style-type: none">• Editor Help menu item (see “The Help Menu” on page 75)• F1 |
|--|

Identify

Waits until you execute another command, either by key presses or mouse clicks, and then displays help for that command instead of executing the command.

By default, this command is bound to:

- | |
|--|
| <ul style="list-style-type: none">• Identify menu item (see “The Help Menu” on page 75) |
|--|

if Conditional Commands

Conditional commands are useful for constructing scripts and key bindings that apply to a particular state or mode in the Editor. For example, **if** commands can be used to make your key bindings respond differently depending on whether the Editor is in insert mode or whether text is selected.

The basic construction of a conditional command is:

if *condition* {*cmds1*} [**else** {*cmds2*}]

If *condition* is true, the commands given for *cmds1* are executed. If *condition* is false, *cmds2* executes.

Specifying the keyword **else** is optional. If **else** is included and the *condition* is not true, then the second command executes.

The following table provides valid values for *condition*.

<code><allowmiddleclick></code>
This condition is true if the configuration option allowMiddleClick is on (for information, see “The MULTI Editor Options Tab” on page 251).
<code><beforenonwhite></code>
This condition is true if there is no selection and the cursor is before all non-whitespace characters on the line (i.e., all characters before the cursor are whitespace).
<code><bsearch></code>
This condition is true if the Editor is in the progress of searching a text backward.
<code><col=eof></code>
This condition is true if the first selected character or the cursor position (if there is no selection) is at the end of the last character (not inclusive) in the corresponding file.
<code><col=eol></code>
This condition is true if the first selected character or the cursor position (if there is no selection) is at the end of the last character (not inclusive) in the corresponding line.
<code><col=sof></code>
This condition is true if the first selected character or the cursor position (if there is no selection) is the first column of a file.

<code><col=sol></code>
This condition is true if the first selected character or the cursor position (if there is no selection) is the first column of a line.
<code><fsearch></code>
This condition is true if the Editor in the progress of searching a text forward.
<code><HaveUnconfirmedAcString></code>
This condition is true if an auto-completed string that was not accepted by the user exists in the Editor.
<code><insertmode></code>
This condition is true if the Editor is in insert mode (see EnterInsertMode in “Mode Commands” on page 295).
<code><lang=language_name></code>
This condition is true if the language used to color the active Editor source file is the one specified. The language name is the name specified in the <code>general</code> section of the corresponding language’s syntax configuration file (see “The language.gsc Syntax Definition Files” on page 49 for details).
<code><noselection></code>
This condition is true if there is no primary selection.
<code><nosselection></code>
This condition is true if there is no secondary selection.
<code><searching></code>
This condition is true if an incremental search is currently in progress, and false otherwise (see iSearch in “Search Commands” on page 301).
<code><select num=line></code>
This condition is true if the selection number <i>num</i> aligns on line boundaries. The primary selection is number 0, and the secondary selection is number 1.
<code><wrapsearch></code>
This condition is true if the Editor in the progress of searching a text in a direction and has wrapped at the boundary.

Example 1. Conditional Commands

```
if <noselection> {SelectLine}; Cut1
```

If there is no selection, the entire line is selected. The **Cut1** command is always executed.

```
if <noselection> {ContinueSelection; SOL} else {Delete}
```

If there is no selection, the current cursor position to the end of the line is selected. If there is a selection, it is deleted.

Indentation Commands

Indentation is whitespace at the beginning of each line that may be used to denote the hierarchical structure of your code more clearly, thus making it more readable. For more information about indents, see “Indenting Code” on page 36.

AutoIndent

Indents the current line or block of lines to the position indicated by the syntax of the code. This command is only available for C, C++, and Ada. Several configuration options affect the operation of **AutoIndent** (see “The MULTI Editor Options Tab” on page 251).

By default, this command is bound to:

- **Auto Indent** menu item (see “The Block Menu” on page 66)
- Ctrl + 2

AutoIndentImplicit

Performs the same function as **AutoIndent**, except that it can be turned off by setting the configuration option **aiimplicitindent** to `off`.

By default, this command is bound to:

- **Auto indent as you type** check box (see “The MULTI Editor Options Tab” on page 251)

AutoIndentOrTab

Performs the same function as **AutoIndent**, except that if the current file is in a language not supported by **AutoIndent**, a Tab is inserted instead.

By default, this command is bound to:

- Tab

Indent

Adds an indent at the beginning of the current line or selection.

By default, this command is bound to:

- **Indent** menu item (see “The Block Menu” on page 66)
- Ctrl + i

Unindent

Removes an indent from the beginning of the current line or selection.

By default, this command is bound to:

- **Unindent** menu item (see “The Block Menu” on page 66)
- Ctrl + Shift + i

Insert Commands

These commands add text to the open file at the cursor’s current position.

"any_string"

Inserts the string between the double quotation marks into the open file at the cursor’s current position. The current selection, if any, is replaced with the text between the quotes. Standard C quoting sequences (for example, \n, \t, \\) are allowed.

AddStr string

Adds the specified *string* to the current cursor location. The *string* replaces the current selection (if any).

Beep

Sounds an audible tone.

Date

Inserts the date and time into the open file at the cursor’s current position.

By default, this command is bound to the menu item:

- **Insert Date** (see “The Tools Menu” on page 68).

InsertFile

Opens the **Insert** dialog box, which allows you to select a file to be inserted at the current line. The contents of the selected file are placed on the line above the cursor.

By default, this command is bound to:

- **Insert File** menu item (see “The Block Menu” on page 66)

InsertNewline

Inserts a newline (`\n`) character into the open file at the cursor's current position. The current selection, if any, is replaced with the newline.

By default, **InsertNewline; AutoIndent** is bound to:

- Enter

By default, **EOL; InsertNewLine** is bound to:

- Ctrl + Shift + R

Tab

Inserts a tab (`\t`) character into the open file at the cursor's current position. The current selection, if any, is replaced with the tab.

UserName

Inserts the current user name (in lowercase letters) into the open file at the cursor's current position.

Mode Commands

The following commands change the Editor's mode.

AlterMode [*mode number*]

Creates bindings for sequences of key combinations, instead of just single-stroke key combinations. This command is typically only bound to keys. For more information about binding commands to keys, see the **keybind** command in "Customizing Keys and Mouse Behavior" on page 191.

The **AlterMode** command switches the Editor to any of 10 modes, called `Edit0` through `Edit9`. In each mode, keystrokes can have different behaviors depending on what bindings have been set for that particular mode. This is useful because any given key combination can have up to 10 meanings, depending on whether or not it was preceded by a key that is bound to the **AlterMode** command.

Usually, you use the **keybind** command to bind a command to a key with modifiers, such as Ctrl or Shift keys. However, with **AlterMode** you can bind commands to a key sequence. The **keybind** command requires the specification of an editing *mode*. When you press the key in that mode, the given command executes. Keys bound to commands in the MULTI Editor are usually bound with the mode set to `Edit` (which refers to the same mode as `Edit0`). However, if this command is specified and the mode is set to `Edit1` – `Edit9`, the next key press in the Editor will execute the command that it is bound to the new mode. For example:

```
keybind "x" | Control@Edit=AlterMode Edit2
keybind "s" | Control@Edit2=SaveFile
```

With this command, pressing Ctrl + X in the Editor changes the Editor mode to `Edit2` for the next key press. Then pressing Ctrl + S in the Editor saves the current file, because the Editor is now in `Edit2` mode.

Any command issued while in an alternate mode will switch the mode back to `Edit0`, the default mode.

EnterInsertMode

Puts the Editor into insert mode, so you can enter literal keystrokes into your file, even if the keys are bound to commands.

For example, suppose you customized the Editor so that every time you press `d`, the cursor moves down one line. This makes it impossible to type the letter `d` in the file. When you turn on insert mode and press `d`, the letter `d` appears in the file, and the cursor does not move down one line.

Insert mode may be turned off by the **FlashCursor** command (see **FlashCursor** in "Navigation Commands" on page 296).

Quote

Forces the character generated by the next key press to be literally entered in the file, even if the key is bound to a command. This is useful for entering characters that have commands bound to them.

By default, this command is bound to:

- Ctrl + \ (backslash)

ToggleErrorView

Toggles *error view mode*, which is off for normal Editor windows. The error view mode allows tag commands to operate properly for output windows. See “Tag Commands” on page 307 for more information.

Navigation Commands

These commands change the location of the cursor, scrolling the open file as necessary to keep the cursor within the Editor’s window. None of these commands modify the contents of the open file.

Column [*column_number*]

Moves the cursor to the specified column in the current line. If no parameter is specified, this command opens a dialog box for you to specify the column to which the cursor should be moved.

By default, this command is bound to:

- **Column** menu item (see “The View Menu” on page 62).

Down

Moves the cursor down one line.

By default, this command is bound to:

- DownArrow
- Ctrl + J

DownSome

Moves the cursor down by *n* lines, where *n* defaults to 5, but may be changed via the **Ctrl+cursor jump size** field located on the **Config** → **Options**→**Editor** tab. For more information, see the **Ctrl+cursor jump size** option in “The MULTI Editor Options Tab” on page 251.

By default, this command is bound to:

- Ctrl + DownArrow

EditLine

Displays the **Goto Line:** prompt, which allows you to move to a specific line by typing in the line number and pressing Enter.

See also the **Goto** and **LineD** commands below.

By default, this command is bound to:

- Ctrl + G

EOF

Moves the cursor to the last column of the last line of the open file (*end of file*).

By default, this command is bound to:

- Ctrl + End

EOL

Moves the cursor to the last column of the current line (*end of line*).

By default, this command is bound to:

- End
- Ctrl + E

FlashCursor

Flashes the line the cursor is on. This command also turns insert mode off (see the **EnterInsertMode** command in “Mode Commands” on page 295).

By default, this command is bound to:

- **Flash Cursor** menu item (see “The View Menu” on page 62)
- The command **NoSelection; FlashCursor** is bound to the Esc key.

Goto

Opens the **GoTo** dialog box. See “Using the GoTo Dialog Box” on page 25 for more information.

By default, this command is bound to:

- **Goto** menu item (see “The Edit Menu” on page 59)
- Ctrl + Shift + G

Left

Moves the cursor one character to the left. If the cursor is already in the first column of the line, this command has no effect.

LeftSome

Moves the cursor to the left by *n* characters, where *n* defaults to 5, but may be changed via the **Ctrl+cursor jump size** field located on the **Config** → **Options**→**Editor** tab. For more information, see the **Ctrl+cursor jump size** option in “The MULTI Editor Options Tab” on page 251.

LeftU

Moves the cursor one character to the left. If the cursor is in the first column of the line, then it will move to the last column of the previous line. If the cursor is already in the first column of the first line of the open file, this command has no effect.

By default, this command is bound to:

- LeftArrow
- Ctrl + H

LineD [*line_number*]

Moves the cursor to the specified line. If no parameter is specified, this command opens a dialog box for you to specify the line to which the cursor should be moved.

MoveCursor [-gui | -string] *line_number*, *column_position*

Moves the cursor to the specified *line_number* and *column_position*. Pass the *-gui* option to specify a GUI column position, or pass the *-string* option to specify a string column position. The GUI and string column positions differ when a Tab character is present. By default, Tab is interpreted as 8 characters in the GUI and as 1 character in a string. The column positions are 0-based; that is, they start at zero.

PageDown

Moves the cursor down one screen.

By default, this command is bound to:

- PageDown
- Ctrl + Shift + N

PageUp

Moves the cursor up one screen.

By default, this command is bound to:

- PageUp
- Ctrl + Shift + B

Return

Moves the cursor to the first column of the next line.

ReverseWord

Moves the cursor to the beginning of the current word. If the cursor is not currently within a word (a group of consecutive alphanumeric characters), it moves to the beginning of the previous word in the open file.

By default, this command is bound to:

- Ctrl + LeftArrow

Right

Moves the cursor one character to the right. If the cursor is already in the last column of the line, this command has no effect.

RightD

Moves the cursor one character to the right. If the cursor is in the last column of the line, then it will move to the first column of the next line. If the cursor is already in the last column of the last line of the open file, this command has no effect.

By default, this command is bound to:

- RightArrow
- Ctrl + L

RightSome

Moves the cursor to the right by *n* characters, where *n* defaults to 5, but may be changed via the **Ctrl+cursor jump size** field located on the **Config** → **Options** → **Editor** tab. For more information, see the **Ctrl+cursor jump size** option in “The MULTI Editor Options Tab” on page 251.

SOF

Moves the cursor to the first column of the first line of the open file (*start of file*)

By default, this command is bound to:

- Ctrl + Home

SOL

Moves the cursor to the immediate left of the first non-whitespace character on the line (*start of line*). If the cursor is already to the left of the first non-whitespace character, then it will move to the first column of the current line.

By default, this command is bound to:

- Ctrl + W

SOLO

Moves the cursor to the first column of the current line (before any indentation) even if the first character is indented.

By default, this command is bound to:

- Home
- Ctrl + 0 (zero)

SOL1

Moves the cursor immediately to the left of the first non-whitespace character on the line, even if the cursor is currently to the left of the first non-whitespace character. If the line has only whitespace characters, the cursor moves to the end of the line.

Up

Moves the cursor up one line.

By default, this command is bound to:

- UpArrow
- Ctrl + K

UpSome

Moves the cursor up by n lines, where n defaults to 5, but may be changed via the **Ctrl+cursor jump size** field located on the **Config** → **Options**→**Editor** tab. For more information, see the **Ctrl+cursor jump size** option in “The MULTI Editor Options Tab” on page 251.

By default, this command is bound to:

- Ctrl + UpArrow

Word

Moves the cursor to the end of the current word. If the cursor is not currently within a word (a group of consecutive alphanumeric characters), it moves to the end of the next word in the open file.

By default, this command is bound to:

- Ctrl + RightArrow

Search Commands

The MULTI Editor supports two methods of searching: incremental searching and interactive searching. For more information about each type of searching, see “Searching in the Editor” on page 28.

BackiSearch

Starts an incremental search that proceeds backward from the current location. See “Incremental Searching” on page 28 for more information.

By default, this command is bound to:

- Ctrl + B

On UNIX, **BackiSearch**; **Search** is bound to:

- Shift + Find or Shift + L9

iSearch

Starts an incremental search that proceeds forward from the current location. See “Incremental Searching” on page 28 for more information.

By default, this command is bound to:

- Ctrl + F

Search

Opens the Search dialog box, which allows you to specify interactive search criteria. See “Interactive Searching Using the Search Dialog Box” on page 29 for more information.

By default, this command is bound to:

- **Find** menu item (see “The Edit Menu” on page 59)
- Ctrl + Shift + F
- Find or L9 (UNIX only)

StopSearch

Stops an incremental or interactive search. The most recently matched or partially matched text will remain selected. See “Incremental Searching” on page 28 for more information.

By default, this command is bound to:

- Esc

TruncateSearch

Restarts the incremental search at the current location. See “Incremental Searching” on page 28 for more information.

Selection Commands

Selection commands can be used to create or cancel the following two types of selections:

- The *primary selection*, which is created by dragging over text with the left mouse button, can be manipulated by many commands, such as clipboard commands, indentation commands, and drag-and-drop commands. The primary selection is replaced by any typed or inserted text and is canceled by any navigation command. One end of the primary selection, the *cursor-end*, is always at the cursor's current location, and may be either the start or the end of the selection.
- The *secondary selection*, which is created by dragging over text with the middle mouse button, can be replaced using either the command **SecondarySelectionReplace** or **SecondarySelectionReplaceClip**. When the middle button is released, **SecondarySelectionReplace** is executed. Any other non-selection Editor command or keystroke cancels a secondary selection if it exists. For more information about the middle mouse button, see **allowMiddleClick** in “The MULTI Editor Options Tab” on page 251.

Many commands operate differently upon selected text. These differences are documented in the individual command descriptions. Commands that do not explicitly mention the secondary selection operate on the primary selection only. With the exception of the commands **SecondarySelectionReplace** and **SecondarySelectionReplaceClip**, none of the commands in this section modify the contents of the open file.

ContinueSelection; Down
Extends the selection down one line. By default, this command is bound to: • Shift + DownArrow
ContinueSelection; DownSome
Extends the selection down a number of lines. By default, this command is bound to: • Ctrl + Shift + DownArrow

ContinueSelection; EOL

Extends the selection to the end of the line.

By default, this command is bound to:

- Shift + End
- Ctrl + Shift + E

ContinueSelection; LeftU

Extends the selection left one character.

By default, this command is bound to:

- Shift + LeftArrow

ContinueSelection; PageDown

Extends the selection down one page.

By default, this command is bound to:

- Ctrl + Shift + PageDown

ContinueSelection; PageUp

Extends the selection up one page.

By default, this command is bound to:

- Shift + PageUp

ContinueSelection; Return

Extends the selection to the beginning of the next line.

On UNIX, this command is bound to:

- Ctrl + Shift + Enter

ContinueSelection; ReverseWord

Extends the selection to the beginning of the current word. If the cursor is not currently in a word, the selection is extended to the beginning of the previous word.

By default, this command is bound to:

- Ctrl + Shift + LeftArrow

ContinueSelection; RightD

Extends the selection right one character.

By default, this command is bound to:

- Shift + RightArrow

ContinueSelection; SOLO

Extends the selection to the beginning of the line.

By default, this command is bound to:

- Shift + Home

ContinueSelection; Up

Extends the selection up one line.

By default, this command is bound to:

- Shift + UpArrow

ContinueSelection; UpSome

Extends the selection up a number of lines.

By default, this command is bound to:

- Ctrl + Shift + UpArrow

ContinueSelection; Word

Extends the selection to the end of the current word. If the cursor is not currently in a word, the selection is extended to the end of the next word.

By default, this command is bound to:

- Ctrl + Shift + RightArrow

GetSelectedString

Outputs the selected string to a client (for example, the **MULTI Python GUI**) or to a dialog box if there is no client.

GetSelection [-gui | -string]

Outputs the current selection range to a client (for example, the **MULTI Python GUI**) or to a dialog box if there is no client. Specify `-gui` to get the GUI range, or specify `-string` to get the string range. The GUI and string ranges differ when a Tab character is present. By default, Tab is interpreted as 8 characters in the GUI and as 1 character in a string.

In the output, line numbers are 1-based; that is, they start at one. Column numbers are 0-based; that is, they start at zero.

NoSelection

Moves the cursor to the beginning of the current primary selection and cancels the selection. If there is no primary selection, this command has no effect.

SecondarySelectAll, SecondarySelectLine, SecondarySelectWord

Identical to their primary selection counterparts, except that these commands operate on the secondary selection.

SecondarySelectionAdjust, SecondarySelectionExtend, SecondarySelectionStart

Identical to their primary selection counterparts, except that these commands operate on the secondary selection.

SecondarySelectionReplace

Deletes the text in the secondary selection, and replaces it with a copy of the text in the primary selection. This command also cancels the secondary selection.

SecondarySelectionReplaceClip

Deletes the text in the secondary selection, and replaces it with a copy of the text in the clipboard. This command also cancels the secondary selection. For more information about the clipboard, see “Clipboard Commands” on page 279.

SelectAll

Selects the entire open file.

By default, this command is bound to:

- **Select All** menu item (see “The Edit Menu” on page 59)
- Ctrl + A
- Quadruple-Left-click

SelectionAdjust, SelectionExtend, SelectionGrab

Create and manipulate selections in a way that is dependent upon the mouse. To be useful, these commands should be bound to mouse buttons and used in conjunction with one another. For a sense of how these commands are used, see “Default Mouse Bindings” on page 336.

SelectionStart

Starts a new primary selection.

By default, this command is bound to:

- Left-click

SelectLine

Selects the current line—including any indentation—and moves the cursor to the end of the line.

By default, this command is bound to:

- Triple-Left-click

SelectMatch

Selects the corresponding paired character if the character immediately following the cursor is a paired character such as a parenthesis, square bracket, quote, or curly brace. If the cursor is not in front of a paired character, or if there is no match, this command has no effect.

SelectRange [-gui | -string] *begin_line, begin_column, end_line, end_column*

Selects a range from *begin_line, begin_column* to *end_line, end_column*. Pass the `-gui` option to specify a GUI column position, or pass the `-string` option to specify a string column position. The GUI and string column positions differ when a Tab character is present. By default, Tab is interpreted as 8 characters in the GUI and as 1 character in a string.

Line numbers start at one. Column numbers start at zero.

SelectToLines

Extends the current selection so that it completely includes the first and last lines of the current selection. The new selection will begin at the first column of the first line of the current selection and will end at the last column of the last line of the current selection. If there is no current selection, this command will select the current line, exactly like the command **SelectLine**.

SelectToMatch

Extends the selection to include the nearest paired characters that enclose the current selection and all text between them. If the cursor is not between paired characters, this command selects the nearest paired characters on either side of the cursor.

By default, this command is bound to:

- **Match** menu item (see “The View Menu” on page 62)
- Shift + Right-click

SelectWord

Selects the current word and moves the cursor to the end of the word.

By default, this command is bound to:

- Double-Left-click

SOLSecondary

Starts the secondary selection at the first column of the current line, and changes the secondary selection to zero length. That is, the secondary selection will contain no characters.

Tag Commands

On UNIX, the MULTI Editor can search tag databases to find the locations of tags (such as procedure names). You can use the various tag commands to quickly jump to and edit these locations. Tag databases are created by programs such as **ctags**. For more information, see “Using Tags in Your Files” on page 27.

AppendTagFile

Appends the entries from the specified tag file into the tag database for later queries.

By default, this command is bound to:

- **Append TagFile** menu item (see “The Tools Menu” on page 68)

ErrorOrTag

Finds the next file/line number combination of an output window and opens the file in a new Editor window at the indicated line number. When the next item is found, the cursor moves to it, so that subsequent uses of this command will open different locations each time.

The output window searched is determined in the following ways:

- If the current window is an output window, then it will be searched.
- If the current window is not an output window, then the last output window searched by this command will be searched again.
- If the current window is not an output window, and there is no output window last searched by this command, then the latest output window will be searched.
- If no output window exists, then this command generates an error.

NewTag

Examines the line where the cursor is located if the window is an output window. If the line contains a file/line number combination, the file is opened in another Editor window at the indicated line number. If the line does not contain a file/line number combination, this command has no effect.

If the window is not an output window, this command uses the currently selected text as a tag name and searches for it in the currently loaded tag database. If the tag is found, the tag location is opened in a new Editor window. If no text is selected, this command has no effect.

By default, the command **SelectWord; NewTag** is bound to:

- Ctrl + T

OpenTag

Examines the line where the cursor is located if the window is an output window. If the line contains a file/line number combination, the file is opened in a new Editor window at the indicated line number. If the line does not contain a file/line number combination, this command has no effect.

If the window is not an output window, this command uses the currently selected text as a tag name and searches for it in the currently loaded tag database. If the tag is found, the tag location is opened in the same Editor window (the new file is pushed onto the file stack, if necessary). If the currently selected text is a paired character (such as a parenthesis or curly brace), the corresponding paired character is selected. If no text is selected, this command has no effect.

ResetTags

Reloads tags from the default tag file (**tags**) and from any additional tag files specified via the **Tools** → **Append TagFile** menu item (see “The Tools Menu” on page 68) or the **AppendTagFile** command (see above).

By default, this command is bound to:

- **Reset Tags** menu item (see “The Tools Menu” on page 68)

SpecialTag

Examines the line where the cursor is located if the window is an output window. If the line contains a file/line number combination, the file is opened in a new Editor window at the indicated line number. If the line does not contain a file/line number combination, this command has no effect.

If the window is not an output window, this command has no effect.

Text Deletion Commands

The following commands delete text in the open file, either at the cursor or in the primary selection. After text is deleted by these commands, it can only be recovered with the **Undo** command. For more information, see “Undo/Redo Commands” on page 312.

Backspace

Deletes the text in the current primary selection and cancels the primary selection. If there is no selection, then this command deletes the character immediately before the cursor.

By default, this command is bound to:

- Backspace key

if <noselection> { ContinueSelection; RightD }; Delete

Deletes the text in the current primary selection and cancels the primary selection. If there is no primary selection, then this command does nothing.

By default, this command is bound to:

- Delete
- Ctrl + D

if <noselection> { ContinueSelection; ReverseWord }; Backspace

Deletes the previous word.

By default, this command is bound to:

- Ctrl + Backspace

if <noselection> { ContinueSelection; SOL }; Cut1

Cuts to the beginning of the line.

By default, this command is bound to:

- Ctrl + U

if <noselection> { ContinueSelection; Word }; Delete

Deletes the next word.

By default, this command is bound to:

- Ctrl + Delete

SelectToLines; Cut1

Cuts an entire line to clipboard number 1.

By default, this command is bound to:

- Ctrl + M

Tools Commands

The Editor provides several file utilities that perform such functions as merging, comparing, and executing shell commands. The following commands provide access to these utilities and to other tools.

! [*command*;] (UNIX only)

Identical to the **ExecuteCmd** below.

CommandToWindow [*command*;] (UNIX only)

Sends the given *command* to the shell and displays the output in a new Editor window.

The parameter must end with a semicolon (;). If no parameters are specified, this command opens the **Command to Window** dialog box.

By default, this command is bound to:

- **Command to Window** menu item (see “The Tools Menu” on page 68)

DiffFiles

Opens the **Diff Files** dialog box, which allows you to find and display the differences between two files. For more information, see “Comparing Files” on page 45.

By default, this command is bound to:

- **Diff Files** menu item (see “The Tools Menu” on page 68)

ExecuteCmd [*command*;] (UNIX only)

Sends the given *command* to the shell as a command and inserts the output into the open file. The parameter must end with a semicolon (;).

If no parameters are specified for this command, it opens the **Shell Command to exe** dialog box.

By default, this command is bound to:

- **Execute Shell Command** menu item (see “The Tools Menu” on page 68)

Grep

Opens the Search in Files dialog box, which allows you to search for an expression in all open files. See “Searching in Files” on page 33 for a description of this window and how to use it.

By default, this command is bound to:

- **Search in Files** menu item (see “The Tools Menu” on page 68)

Make (UNIX only)

Opens the **Thing to make?** dialog box.

By default, this command is bound to:

- **Make** menu item (see “The Tools Menu” on page 68)

MergeFiles

Opens the **Merge** dialog box to merge two or three files. For more information, see “Merging Files” on page 40.

By default, this command is bound to:

- **Merge Files** menu item (see “The Tools Menu” on page 68)

Minibuffer

Opens the **Execute Editor Commands** dialog box, which allows you to enter commands.

By default, this command is bound to:

- **Execute Editor Commands** menu item (see “The Tools Menu” on page 68)

MultiBar

Opens the MULTI Launcher.

Notepad

Opens a small Editor window on a scratch file.

By default, this command is bound to:

- **Notepad** menu item (see “The Tools Menu” on page 68)
- Ctrl + Shift + Right-click

Shell [*command*;]

Sends the given command to the shell as a command. The parameter must end with a semi-colon (;). The output is sent to the console from which MULTI was launched. This command does not modify the open file.

If no parameters are specified, this command opens the **Shell Command to execute** dialog box.

WGUtils

Opens the **Utility Program Launcher** dialog box, which allows you to launch utilities.

By default, this command is bound to:

- **Launch Utility Programs** menu item (see “The Tools Menu” on page 68)

Undo/Redo Commands

The following commands allow you to undo, repeat, or cancel other commands and actions that have already executed or are currently executing.

Abort

Aborts any ongoing command, such as a search.

By default, this command is bound to:

- Esc

Redo

Restores the edit that was removed by **Undo**.

By default, this command is bound to:

- **Redo** menu item (see “The Edit Menu” on page 59)
- Ctrl + Y
- Again or L2 (UNIX only)

RepeatLast

Repeats the last edit made to the file.

By default, this command is bound to:

- **Repeat Last Edit** menu item (see “The Edit Menu” on page 59)
- Ctrl + . (period)

ShowContextMenu

Opens a context-sensitive shortcut menu at the cursor. This command should only be bound to mouse buttons.

By default, this command is bound to:

- Right-click

Undo

Reverses the last change made to the current file.

By default, this command is bound to:

- **Undo** menu item (see “The Edit Menu” on page 59)
- Ctrl + Z
- Undo or L4 (UNIX only)

Version Control Commands

The following commands are used with version control.

AllowAutoCheckout

Enables auto checkout from version control. This only affects buffers that are under version control.

By default, this command is bound to:

- **Auto Checkout** menu item (see “The Version Menu” on page 71)

CheckIn

Checks the file into version control, prompting you for comments.

By default, this command is bound to:

- **Check In/Commit** menu item (see “The Version Menu” on page 71)

CheckOut

Checks the file out of version control.

By default, this command is bound to:

- **Check Out** menu item (see “The Version Menu” on page 71)

Discard

Discards the current checkout.

By default, this command is bound to:

- **Discard Changes** menu item (see “The Version Menu” on page 71)

PlaceUnderVC

Places the current file under version control.

By default, this command is bound to:

- **Place Under VC** menu item (see “The Version Menu” on page 71)

PreventAutoCheckout

Opposite of **AllowAutoCheckout**.

By default, this command is bound to:

- **Auto Checkout** menu item (see “The Version Menu” on page 71)

QuerySaveCheckinAll

Saves the changes you have made to all the files you have checked out, makes the files read only in MULTI, and removes the lock from the files. You will be asked for comments to be saved in the log file along with your changes.

By default, this command is bound to:

- **Check In All** menu item (see “The Version Menu” on page 71)

RevertDate

Displays a dialog box which allows you to enter a version date.

By default, this command is bound to:

- **Revert to Date** menu item (see “The Version Menu” on page 71)

RevertHistory

Opens a dialog box which allows you to select a version to revert the file to.

By default, this command is bound to:

- **Revert to History** menu item (see “The Version Menu” on page 71)

RevertToBackup

Reverts the current file to the backup file. This command is only valid if the **editorBackups** configuration option is set to `on`. For more information, see “The MULTI Editor Options Tab” on page 251.

By default, this command is bound to:

- **Revert to Backup** menu item (see “The File Menu” on page 56)

RevertVersion

Opens a dialog box to enter a version number to load into the Editor.

By default, this command is bound to:

- **Revert to Version** menu item (see “The Version Menu” on page 71)

ShowHistory

Opens a dialog box that displays the version control history.

By default, this command is bound to:

- **Show History** menu item (see “The Version Menu” on page 71)
- `Ctrl + Shift + H`

ShowLastEdit

Finds the version of the current file that changed the selected text. This command functions the same as the GUI menu item (see “The Version Menu” on page 71).

By default, this command is bound to:

- **Show Last Edit** menu item (see “The Version Menu” on page 71)
- Ctrl + Shift + L

ShowView

Displays the user’s current view.

Note: This command is only supported when using ClearCase.

VCBuffer

Opens a dialog box that prompts you for a version control command. The full name of the current file is appended at the end of the command.

View Commands

The following table lists view commands.

BrowseObjXRef

Displays the object’s cross references (if any) in a Browse window. This command is available for use with C, C++, and Ada files. For more information, see “Browse References” on page 24.

By default, this command is bound to:

- **Browse References of Selected Object** menu item (see “The View Menu” on page 62).

CloseDependentWindows

Deletes all cross reference Browse windows.

By default, this command is bound to:

- **Close Dependent Windows** menu item (see “The View Menu” on page 62).

CyclePush

Accesses the previous open file in the Editor’s stack.

By default, this command is bound to:

- **Previous File** menu item (see “The View Menu” on page 62)
- Ctrl + Tab

CyclePushBack

Accesses the next open file in the **Editor's** stack.

By default, this command is bound to:

- **Next File** menu item (see “The View Menu” on page 62)
- Ctrl + Shift + Tab

DrawWrapLine [On | Off | AsWordWrap]

Specifies whether or not to draw the wrap line in the MULTI Editor. Permitted settings for this command are:

- On — [default] Enables the wrap line.
- Off — Disables the wrap line.
- AsWordWrap — Enables the wrap line if word wrap is enabled.

This command does the same thing as the configuration option **drawWrapLine**.

GenerateXrefInfo

Obtains complete cross reference information based on the project to which the source file belongs. This command is available for use with C and C++ files.

After cross reference information for the enclosing project is generated, use the **RegenerateXrefInfo** command.

By default, this command is bound to:

- **Generate Cross References** menu item (see “The View Menu” on page 62).

GotoObjDecl

Displays the source code (if available) for the object's declaration in the source pane. This command is available for use with C, C++, and Ada files.

By default, this command is bound to:

- **Go To Declaration of Selected Object** menu item (see “The View Menu” on page 62).

GotoObjDef

Displays the source code (if available) for the object's definition in the source pane. This command is available for use with C, C++, and Ada files.

By default, this command is bound to:

- **Go To Definition of Selected Object** menu item (see “The View Menu” on page 62).

NextWindow

Moves the cursor into the next Editor window and raises that window to the foreground. Using this command repeatedly will eventually cycle through all of the open Editor windows.

RegenerateXrefInfo

Regenerates cross reference information based on the project to which the source file belongs. This command is available for use with C and C++ files.

This command is used after **GenerateXrefInfo** has already been used to generate cross reference information for the enclosing project.

By default, this command is bound to:

- **Regenerate Cross References** menu item (see “The View Menu” on page 62).

Deprecated Commands

The following table lists deprecated commands.

AlterLocation

Use the **AlterMode** command instead (see **AlterMode** in “Mode Commands” on page 295).

CmdPrompt2Wnd

Use the **CommandToWindow** command instead (see **CommandToWindow** in “Tools Commands” on page 310).

CreateLog

Use the **PlaceUnderVC** command instead (see **PlaceUnderVC** in “Version Control Commands” on page 313).

EditTag

Use the **OpenTag** and **NewTag** commands instead (see **OpenTag** and **NewTag** in “Tag Commands” on page 307).

OpenText

Use the **OpenFile** command instead (see **OpenFile** in “File Commands” on page 285).

Third-Party Tools

Appendix B

This Appendix Contains:

- Third-Party Version Control Systems
- Third-Party Editors
- Using the Editor with Third-Party Tools

This chapter describes how to use various third-party tools with the MULTI Editor.



Note For information about using third-party compilers and using the MULTI Debugger with third-party tools, see Appendix D, “Using Third-Party Tools with the MULTI Debugger” in the *MULTI: Debugging* book.

Third-Party Version Control Systems

MULTI can be configured to integrate with the following third-party version control systems:

- ClearCase (see “Integrating with ClearCase” on page 128).
- CVS (see “Integrating with CVS” on page 129.)
- PVCS (see “Integrating with PVCS” on page 130).
- RCS (see “Integrating with RCS” on page 131).
- SourceSafe (see “Integrating with SourceSafe” on page 132).
- Subversion (see “Integrating with Subversion” on page 134).

Third-Party Editors

You can configure MULTI to use another editor in place of MULTI’s built-in Editor. To specify an alternate editor, you can either set options using the **Other Editor Configuration** dialog box or you can use the appropriate configuration options. For instructions on how to specify an alternate editor, see “Third-Party Editor Configuration Options” on page 213.

Using the Editor with Third-Party Tools

You can configure the Editor to launch third-party tools from buttons or menus. The following Editor commands are useful for this purpose:

! *[command;]* (UNIX only)

Takes the current selection in the Editor, if any, and pipes it to the standard input of the specified command. If there is a selection in the Editor, the selection is replaced with the output of the specified command. If there is no selection, the command receives no input, and the output is inserted at the cursor. This is equivalent to the **ExecuteCmd** command.

CommandToWindow *[command;]* (UNIX only)

Runs a shell command and opens a new Editor window to which the standard output of the command is redirected.

Shell *[command;]*

Runs a shell command with no input/output redirection.

For more information about these commands, see Appendix A.

Here is an example configuration file that runs some simple UNIX commands:

```
menu: MyTools { \
    {'insert pretty date' ! date "+%l:%M %p on %B %e, %Y"} \
    {'count words' CommandToWindow \
        "printf %8s%8s%8s%9s\\n lines words chars filename ; wc %FILE"}}

menu: EditMenuBar { \
    {&File ->EditFile} \
    {&Edit ->EditEdit} \
    {V&iew ->EditView} \
    {&Block ->EditBlock} \
    {&Tools ->EditTools} \
    {MyTools ->MyTools} \
    {&Version ->EditVersion} \
    {&Config ->Config} \
    {&Windows ->WindowsMenu} \
    {&Help ->EditHelp}}
```



Note The backslashes at the end of lines indicate a line wrap. In the actual configuration file, there should not be any line wraps.

To configure the Editor menus from the GUI, choose **Config → Options**, choose the **General** tab, and click the **Menus** button.

Default Key and Mouse Bindings

Appendix C

This Appendix Contains:

- Default Keyboard Shortcuts
- Default Mouse Bindings

This chapter lists the default key and mouse bindings. For information about how you can change key and mouse bindings, see “Customizing Keys and Mouse Behavior” on page 191 in Chapter 8, “Configuring and Customizing MULTI”.

Default Keyboard Shortcuts

The following tables list default key bindings. You can create or change your key bindings with the **keybind** command (see “Customizing Keys and Mouse Behavior” on page 191).

Navigating

The following key bindings control cursor movement in MULTI windows or navigation through buffers. For more information about the commands, see “Navigation Commands” on page 296.

UpArrow Ctrl + K Moves the cursor up one line. By default, these keyboard shortcuts are bound to the Up command.
Ctrl + UpArrow Moves the cursor up multiple lines [default is 5]. By default, this keyboard shortcut is bound to the UpSome command.
PageUp Ctrl + Shift + B Moves the cursor up one screen By default, these keyboard shortcuts are bound to the PageUp command.
DownArrow Ctrl + J Moves the cursor down one line. By default, these keyboard shortcuts are bound to the Down command.
Ctrl + DownArrow Moves the cursor down multiple lines [default is 5]. By default, these keyboard shortcuts are bound to the DownSome command.

PageDown Ctrl + Shift + N Moves the cursor down one screen By default, these keyboard shortcuts are bound to the PageDown command.
LeftArrow Ctrl + H Moves the cursor left one character. By default, these keyboard shortcuts are bound to the LeftU command.
Ctrl + LeftArrow Moves the cursor to the previous word. By default, this keyboard shortcut is bound to the ReverseWord command.
Ctrl + W Ctrl + ^ Moves the cursor to the first character of the line. By default, these keyboard shortcuts are bound to the SOL command.
Home Ctrl + 0 (zero) Moves the cursor to the beginning of the line. By default, these keyboard shortcuts are bound to the SOL0 command.
Ctrl + Home Moves the cursor to the beginning of the file. By default, this keyboard shortcut is bound to the SOF command.
RightArrow Ctrl + L Moves the cursor right one character. By default, these keyboard shortcuts are bound to the RightD command.
Ctrl + RightArrow Moves the cursor to the next word. By default, this keyboard shortcut is bound to the Word command.

End Ctrl + E Moves the cursor to the end of line. By default, these keyboard shortcuts are bound to the EOL command.
Ctrl + Enter Moves the cursor to the beginning of the next line By default, this keyboard shortcut is bound to the EOL; RightD command.
Ctrl + End Moves the cursor to the end of the file. By default, this keyboard shortcut is bound to the EOF command.
Ctrl + G Prompts for a new line number to go to. By default, this keyboard shortcut is bound to the EditLine command.
Ctrl + Tab Cycles through open file buffers forward. By default, this keyboard shortcut is bound to the CyclePush command. For more information about the command, see “View Commands” on page 315.
Ctrl + Shift + Tab Cycles through open file buffers backward. By default, this keyboard shortcut is bound to the CyclePushBack command. For more information about the command, see “View Commands” on page 315.

Opening, Saving, and Closing

The following keyboard shortcuts are used to open, save, and close files. For more information about the commands, see “File Commands” on page 285.

Ctrl + O Open or L7 (UNIX only) Opens a dialog box to open a file in the current window. By default, this keyboard shortcut is bound to the OpenFile command.
Ctrl + N Ctrl + Shift + O (letter o) Opens a dialog box to open a file in a new window. By default, this keyboard shortcut is bound to the LoadFile command.
Ctrl + S Saves the current file. By default, this keyboard shortcut is bound to the Save command.
Ctrl + Shift + S Opens a dialog box to save the current file. By default, this keyboard shortcut is bound to the SaveAs command.
Ctrl + Shift + Q Saves the current file, then closes the window. By default, this keyboard shortcut is bound to the Done command.
Ctrl + Q Closes the current editing window. By default, this keyboard shortcut is bound to the Close command.

Undo/Redo

The following keyboard shortcuts are used to undo and redo edits. For more information about the commands, see “Undo/Redo Commands” on page 312.

Ctrl + Z Undo or L4 (UNIX only) Reverts the last edit. By default, these keyboard shortcuts are bound to the Undo command.
Ctrl + Y Again or L2 (UNIX only) Reverts the last undo. By default, these keyboard shortcuts are bound to the Redo command.

Cutting, Copying, and Pasting

The following keyboard shortcuts are used to copy, cut, and paste text to and from clipboards. For more information about the commands, see “Clipboard Commands” on page 279.

Ctrl + X Cut or L10 (UNIX only) Cuts the selected text to the clipboard. By default, these keyboard shortcuts are bound to the Cut1 command.
Ctrl + Shift + X Shift + Cut or Shift + L10 (UNIX only) Cuts the selected text to the second buffer. By default, these keyboard shortcuts are bound to the Cut2 command.
Meta + Ctrl + X (UNIX only) Ctrl + Cut or Ctrl + L10 (UNIX only) Cuts the selected text to the third buffer. By default, these keyboard shortcuts are bound to the Cut3 command.

Meta + Ctrl + Shift + X (UNIX only) Ctrl + Shift + Cut or Ctrl + Shift + L10 (UNIX only) Cuts the selected text to the fourth buffer. By default, these keyboard shortcuts are bound to the Cut4 command.
Ctrl + C Copy or L6 (UNIX only) Copies the selected text to the clipboard. By default, these keyboard shortcuts are bound to the Copy1 command.
Ctrl + Shift + C Shift + Copy or Shift + L6 (UNIX only) Copies the selected text to the second buffer. By default, these keyboard shortcuts are bound to the Copy2 command.
Meta + Ctrl + C (UNIX only) Ctrl + Copy or Ctrl + L6 (UNIX only) Copies the selected text to the third buffer. By default, these keyboard shortcuts are bound to the Copy3 command.
Meta + Ctrl + Shift + C (UNIX only) Ctrl + Shift + Copy or Ctrl + Shift + L6 (UNIX only) Copies the selected text to the fourth buffer. By default, these keyboard shortcuts are bound to the Copy4 command.
Ctrl + V Paste or L8 (UNIX only) Pastes from the clipboard. By default, these keyboard shortcuts are bound to: if <noselection> { if <select1=lines> { SOL0 } } Paste1.
Ctrl + Shift + V Shift + Paste or Shift + L8 (UNIX only) Pastes from the second buffer. By default, these keyboard shortcuts are bound to: if <noselection> { if <select1=lines> { SOL0 } } Paste2.

Meta + Ctrl + V (UNIX only) Ctrl + Paste or Ctrl + L8 (UNIX only) Pastes from the third buffer. By default, these keyboard shortcuts are bound to: if <noselection> { if <select1=lines> { SOL0 } } Paste3.
Meta + Ctrl + Shift + V (UNIX only) Pastes from the fourth buffer. By default, these keyboard shortcuts are bound to: if <noselection> { if <select1=lines> { SOL0 } } Paste4.

Deleting Text

The following keyboard shortcuts are used to delete text. For more information about the commands, see “Text Deletion Commands” on page 309.

Backspace Deletes the previous character or the selection. By default, this keyboard shortcut is bound to the Backspace command.
Ctrl + Backspace Deletes the previous word or the selection. By default, this keyboard shortcut is bound to: if <noselection> { ContinueSelection; ReverseWord }; Backspace.
Delete Ctrl + D Deletes the next character or the selection. By default, these keyboard shortcuts are bound to: if <noselection> { ContinueSelection; RightD }; Delete.
Ctrl + Delete Deletes the next word or the selection. By default, this keyboard shortcut is bound to: if <noselection> { ContinueSelection; ReverseWord }; Backspace.
Ctrl + U Cuts to the beginning of the current line or the selection. By default, this keyboard shortcut is bound to: if <noselection> { ContinueSelection; SOL }; Cut1.

Ctrl + M

Cuts the current line or the lines in the selection to the first buffer.

By default, this keyboard shortcut is bound to: **SelectToLines; Cut1**.

Ctrl + P

Joins the current line with the next or the lines in the selection.

By default, this keyboard shortcut is bound to the **JoinLines** command.

Selecting Text

The following key bindings control how text is selected. For more information about the commands, see “Selection Commands” on page 302.

Shift + UpArrow

Extends the selection up one line.

By default, this keyboard shortcut is bound to **ContinueSelection; Up**.

Ctrl + Shift + UpArrow

Extends the selection up multiple lines [default is 5].

By default, this keyboard shortcut is bound to: **ContinueSelection; PageUp**.

Shift + PageUp

Extends the selection up one screen.

By default, this keyboard shortcut is bound to: **ContinueSelection; UpSome**.

Shift + DownArrow

Extends the selection down one line.

By default, this keyboard shortcut is bound to: **ContinueSelection; Down**.

Ctrl + Shift + DownArrow

Extends the selection down multiple lines [default is 5].

By default, this keyboard shortcut is bound to: **ContinueSelection; DownSome**.

Ctrl + Shift + PageDown

Extends the selection down one screen.

By default, this keyboard shortcut is bound to: **ContinueSelection; PageDown**.

Shift + LeftArrow

Extends the selection left one character.

By default, this keyboard shortcut is bound to: **ContinueSelection; LeftU**.

Ctrl + Shift + LeftArrow Extends the selection to previous word. By default, this keyboard shortcut is bound to: ContinueSelection; ReverseWord.
Shift + Home Extends the selection to beginning of line. By default, this keyboard shortcut is bound to: ContinueSelection; SOL0.
Shift + RightArrow Extends the selection right one character. By default, this keyboard shortcut is bound to: ContinueSelection; RightD.
Ctrl + Shift + RightArrow Extends the selection to next word. By default, this keyboard shortcut is bound to: ContinueSelection; Word.
Shift + End Extends the selection to end of line. By default, these keyboard shortcuts are bound to: ContinueSelection; EOL.
Ctrl + Shift + Enter Extends the selection to beginning of next line. By default, this keyboard shortcut is bound to: ContinueSelection; Return.
Ctrl + A Selects the entire document. By default, this keyboard shortcut is bound to: SelectAll.

Searching

The following key bindings are used to control searches. For more information about the commands, see “Search Commands” on page 301.

Ctrl + F Starts or continues an incremental search forward. By default, this keyboard shortcut is bound to the iSearch command.
Ctrl + B Starts or continues an incremental search backward. By default, this keyboard shortcut is bound to the BackiSearch command.

Esc

Cancels a search.

By default, this keyboard shortcut is bound to the **Abort** command.

Ctrl + Shift + F

Find or L9 (UNIX only)

Opens the Search dialog box.

By default, these keyboard shortcuts are bound to the **Search** command.

Shift + Find or Shift + L9 (UNIX only)

Starts an incremental search backwards, then opens the Search dialog box.

By default, these keyboard shortcuts are bound to: **BackiSearch; Search**.

Auto-Completion

The following keyboard shortcuts to perform auto-completion functions, even when auto-completion is disabled.

For information about auto-completion commands, see “Auto-Completion Commands” on page 275

Ctrl +]

Auto-completes the characters with the next string that matches the pattern.

By default, this keyboard shortcut is bound to the **NextAutoCompleteString** command.

Ctrl + [

Auto-completes the characters with the previous string that matched the pattern.

By default, this keyboard shortcut is bound to the **PrevAutoCompleteString** command.

Ctrl + /

Displays a list of matched strings. You can select a string from the list.

By default, this keyboard shortcut is bound to the **AutoCompleteList** command.

Ctrl + '

Displays a list of matched function prototypes. You can select a function prototype from the list.

By default, this keyboard shortcut is bound to the **AutoCompletePrototypeList** command.

Tab

Accepts the string auto-completed by the MULTI Editor, moves the cursor to the end of the selected text, then clears the selection on the text.

Indenting Text

The following keyboard shortcuts are used to indent your text. For more information about the commands, see “Indentation Commands” on page 292.

Ctrl + i

Indents the current line or the selection.

By default, this keyboard shortcut is bound to the **Indent** command.

Ctrl + Shift + i

Unindents the current line or the selection.

By default, this keyboard shortcut is bound to the **Unindent** command.

Tab

Auto indents the current line or the selection.

By default, this keyboard shortcut is bound to: **if <searching> { Tab } else { if <beforenonwhite> { AutoIndentOrTab } else {if <noselection> {Tab} else {AcceptAcString; MoveToEndOfSelectionAndUnselect} } };**

Version Control

The following keyboard shortcuts are used to access version control options. For more information about the commands, see “Version Control Commands” on page 313.

Ctrl + Shift + L

Opens the MULTI Diff Viewer on the last edit to the current selection.

By default, this keyboard shortcut is bound to the ShowLastEdit command.

Ctrl + Shift + H

Opens a version history browser.

By default, this keyboard shortcut is bound to the ShowHistory command.
--

Miscellaneous

The following are miscellaneous keyboard shortcuts.

Ctrl + \

Enters the next key press sequence literally.

By default, this keyboard shortcut is bound to the Quote command.
--

Ctrl + .

Repeats the last command.

By default, this keyboard shortcut is bound to the RepeatLast command.

Ctrl + Shift + A

When entered in a comment, the comment text will be reformatted to wrap across multiple lines.
--

By default, this keyboard shortcut is bound to: FillParagraph .
--

Ctrl + Shift + R

Inserts a new line at the end of the current line.
--

By default, this keyboard shortcut is bound to: EOL; InsertNewLine .

Ctrl + Shift + T

Transposes the previous two characters.

By default, this keyboard shortcut is bound to: NoSelection; ContinueSelection; Left; Cut2; Left; Paste2; Right .
--

Ctrl + T
Edits the definition of the procedure containing the cursor when cross reference information is available. By default, this keyboard shortcut is bound to: SelectWord; NewTag .
Enter
Inserts a new line. By default, this keyboard shortcut is bound to: InsertNewline;AutoIndent .
Esc
Cancels the current selection and flashes the line containing the cursor. By default, this keyboard shortcut is bound to: NoSelection; FlashCursor .

Default Mouse Bindings

The following tables list default mouse bindings. While mouse bindings may be assigned to up to five mouse buttons, these tables assume that your mouse has only three buttons: left (**button1**), middle (**button2**), and right (**button3**). For information about changing mouse bindings, see the **mouse** command, in “Customizing Keys and Mouse Behavior” on page 191.

First (Left) Mouse Button

The following are default mouse bindings for the left mouse button (**button1**). For more information about the commands, see “Selection Commands” on page 302.

[Left-click]
Starts new (primary) selection. By default, this mouse action is bound to the SelectionStart command.
[Left-click] + Drag
Selects text for the primary selection
Shift + [Left-click]
Extends the primary selection to the mouse pointer.
Ctrl + [Left-click]
Copies selection when in drag-and-drop mode.

[Double Left-click]

Selects the current word.

By default, this mouse action is bound to the SelectWord command.
--

[Triple Left-click]

Selects the current line.

By default, this mouse action is bound to the SelectLine command.
--

[Quadruple Left-click]

Selects the entire file.

By default, this mouse action is bound to the SelectAll command.

Second (Middle) Mouse Button

The following are default mouse bindings for the middle mouse button (**button2**). The middle button may not be available, depending on your mouse and the **Allow middle click to paste** configuration option (for more information see “The MULTI Editor Options Tab” on page 251).

[Middle-click]

Makes a secondary selection and replaces it with the primary selection.

[Double Middle-click]

Replaces a current word with the primary selection.

[Triple Middle-click]

Replaces a current line with the primary selection.

[Quadruple Middle-click]

Replaces an entire file with the primary selection.

Third (Right) Mouse Button

The following are default mouse bindings for the right mouse button (**button3**).

[Right-click]	Opens the shortcut menu for performing operations on the current object.
Ctrl + [Right-click]	Edits the definition of the selected procedure when cross reference information is available.
Shift + [Double Right-click]	Selects a matching curly brace { }, square bracket [], or parenthesis (). By default, this mouse action is bound to the SelectToMatch command.
Shift + [Right-click]	If the cursor is located inside of, or immediately outside of, a pair of curly braces { }, square brackets [], or parentheses (), selects everything inside of and including the curly braces, square brackets, or parentheses.
Ctrl + Shift + [Right-click]	Opens a temporary editor for taking notes. By default, this mouse action is bound to the notepad command.

Index

- * (asterisk) wildcard character, 31
- ! command, 310, 321
- > (minus greater than) menu command, 190
- # (number sign)
 - inserting in comments, 255
- ? wildcard character, 31
- “ ” (quotation marks-double) command, 293
- 1, 2, 3, 4, 5, 6, 7, 8 menu item, 58

A

- Abort** command, 312
- About** command, 76, 289
- About MULTI** menu item, 76
- AcceptAcString** command, 275
- action sequences, *see* workspace action sequences
- action tree, 97
- actions, *see* workspace actions
- Ada paren mode** configuration
 - option, 256
- adaContinuationSize** configuration
 - option, 258
- adaindentSize** configuration option, 258
- Add New Files** menu item, 153
- address offsets
 - save between sessions, 262
- AddStr**
 - command, 293
- Again key (UNIX), 312, 328
- aiAdaParenindentMode** configuration
 - option, 256
- aiCharsLikeStarinComment**
 - configuration option, 255
- aiCommentsStayFlushLeft** configuration
 - option, 255
- aiimplicitindent** configuration option, 253
- aiimplicitindentinComments**
 - configuration option, 254
- aiimplicitOnlyAtinitial** configuration
 - option, 254
- aiParenindentMode** configuration
 - option, 256
- aiSwitchinTwo** configuration option, 254
- aiTouchComments** configuration
 - option, 254
- alignment, window, 199
 - See also* window docking
- Allow beeping** configuration option, 209
- Allow middle click to paste** configuration
 - option, 252
- Allow overtype mode** configuration
 - option, 253
- AllowAutoCheckout** command, 72, 313
- allowExecutionin...** configuration option (deprecated), 270
- allowExecutioninBpCommand**
 - configuration option, 241
- allowProcCallinOsaTask** configuration
 - option, 241
- AlterLocation** Editor command (deprecated), 317
- AlterMode** Editor command, 295
- alternative editor configuration, 213

Index

alwaysUseMeToFixBuildErrors

configuration option, 227

“*any_string*” command, 293

AppendTagFile

command, 27, 70, 307

menu item, 70

Application configuration option, 220

Application Options tab, 218

application_name menu item, 74

Ask before halting to set breakpoint

configuration option, 230

assembly_color configuration option, 267

assertionSyntaxChecking configuration

option (deprecated), 270

attemptToShowOldVersion...

configuration option, 241

Auto Checkout menu item, 72

auto indent

Ada, 256

C chars, 255

C paren, 256

comments, 254

comments with multiple lines, 254

implicit, 253 to 254

overview, 36

two levels, 254

Auto indent as you type configuration

option, 253

Auto indent comments as you type

configuration options, 254

Auto Indent menu item, 66

auto-complete

best match, 53, 62

configuration, 52

enable, 62

first match, 53, 62

function prototypes, 53

keyboard shortcuts, 333

maximum match, 275

maximum number of objects, 62

minimum string, 53

minimum string length, 62

auto-detect version control type, 211

AutoCompleteList command, 275

AutoCompletePrototypeList

command, 275

autoConnectionMode configuration

option, 229

autoDwarf2Dbo configuration

option, 237

autoEditErrors configuration option, 226

autoEditWarnings configuration

option, 227

autoGrabHeadFiles configuration

option, 259

AutoIndent command, 66, 292

AutoIndentImplicit command, 292

AutoIndentOrTab command, 292

automatic checkout, 126

ClearCase, 215

configuration option, 215

PVCS, 215

RCS, 215

SourceSafe, 215

Automatically dereference pointers

configuration option, 234

Automatically open editor on errors

configuration option, 226

Automatically save “User Methods”

changes configuration option, 263

Automatically save changed connection

files configuration option, 262

Automatically share target connections

configuration option, 229

Automatically verify ROM image is

up-to-date configuration option, 236

autoSaveConnectionsInFiles

configuration option, 262

autoSaveUserConnections configuration

option, 263

autoStabs2Dbo configuration option, 237

Index

autoVerifyRomSections configuration
option, 236

B

background *color* configuration
option, 266

backgroundMode configuration
option, 231

BackiSearch command, 301

Backspace

command, 309

key, 309, 330

Backward radio button, 30

bDotColor *color* configuration option, 267

beep

command, 293

configuration option, 209

best match auto-completion, 53

bindings

key, 324

mouse, 336

blinkingCursor configuration option, 221

Block menu, 66

blockRun configuration option, 242

blockStep configuration option, 235

Bookmarks menu item, 75

Bookmarks submenu, 75

Both Numbers configuration option, 231

bpSyntaxChecking configuration
option, 234

braces, curly {}, 63

brackets, square [], 63

Break Dot configuration option, 267

breakColor *color* configuration
option, 267

breakpoints

configuration, 243

overwrite scripts, 261

save between sessions, 261

warning, 251

Browse References menu item, 22, 24

Browse References of Selected Object
menu item, 64

Browse window, 24

global scope configuration, 243

BrowseObjXRef

command, 64

BrowseObjXRef command, 315

browser colors, 249

buffers, 311

copying to, 279

cutting to, 280

pasting from, 281

BugReport command, 289

Build Details window

errors, 226

progress, 226

warnings, 227

builderPosition configuration option
(deprecated), 270

buildFilesDir configuration option, 264

buildFileSet configuration option, 264

buildType configuration option, 227

buttons

attaching a menu to, 189

configuring, 242

toolbar, 20

C

C chars aligned like '*' in comments
configuration option, 255

C paren indent mode configuration
option, 256

case-sensitive

language setting, 50

searches, 209

searching, 30, 33

change indicator, 22

character *color* configuration option, 268

characters, inserting in Editor, 197

Index

- Check In All** menu item, 71
- Check Out**
 - menu item, 71
- Check syntax of breakpoints when they are set** configuration option, 234
- CheckIn**
 - command, 71, 313
 - menu item, 71
- checkout
 - create, 150
 - modify, 151
 - status, 142
- Checkout**
 - command, 71, 313
- Checkout Browser
 - Add** menu item, 148
 - Check In** menu item, 147
 - Check Out** menu item, 148
 - Close** menu item, 146
 - Commit** menu item, 147
 - Config** menu, 146
 - Conflicts** menu item, 146
 - CVS Checkout Editor** menu item, 146
 - Delete** menu item, 148
 - Diff with Local Version** menu item, 147
 - Diff with Remote Version** menu item, 147
 - Edit** menu item, 147
 - File** menu, 145
 - Filename** field, 142
 - Full Rescan** menu item, 145
 - Halt Scan / Update** menu item, 146
 - Local Changes** menu item, 146
 - Local Rescan** menu item, 145
 - Local Version** field, 143
 - Locked By** field, 143
 - menus, 145
 - New Window** menu item, 145
 - No changes** menu item, 146
 - Not under VC** menu item, 146
 - overview, 138
 - Place Under VC** menu item, 148
 - Remote Changes** menu item, 146
 - Rescan** menu item, 147
 - Resolve Conflict** menu item, 148
 - Revert to Repository** menu item, 148
 - Roots** field, 139
 - shortcut menu, 147
 - Show History** menu item, 147
 - Show** menu, 146
 - starting, 139
 - Status** field, 142
 - Tag** field, 143
 - Unlock** menu item, 148
 - Update Checkout** menu item, 145
 - Update** menu item, 147
- Clear User Default Configuration**, 179
 - menu item, 74
- clearButtons** configuration option, 242
- ClearCase version control
 - automatic checkout, 215
 - configuration, 124, 211
 - integration, 128
 - path, 215
- ClearConfig** command, 74, 282
- clearEditButtons**
 - configuration option, 259
- clearKeys** configuration option, 221
- clearMenus** configuration option, 221
- clearMice** configuration option, 221
- clickPause** configuration option, 221
- clipboard
 - copying to, 279
 - cutting to, 59, 280
 - keyboard shortcuts, 328
 - manager, 209
 - pasting from, 281
- clipManLaunch** configuration option, 209
- close buttons, displaying on toolbar, 208
- Close** command, 58, 285

Index

Close Dependent Windows menu
 item, 64

Close Editor menu item, 58

Close File menu item, 58

Close menu item, 153

closeButtonOnTitlebar configuration
 option, 208

CloseDependentWindows command, 64,
 315

CmdPrompt2Wnd command
 (deprecated), 317

code, indenting, 36

Color C++ comments in C configuration
 option, 268

color chooser, 269

Coloring for multiple debuggers
 configuration option, 231

Colors

 choosing, 269

 configuration options, 265

 tab, 265

colorSyntax configuration option, 267

Column

 command, 63, 296

 menu item, 63

column of text

 copying, 39

 cutting, 40

 pasting, 40

command help, 6

command line

 configuration file, 177

 script file, 180 to 181

Command line arguments configuration
 option, 214

command line options

-diff, 17

file, 17

-h, 17

-help, 17

 +*line*, 17

-new, 17

-reuse, 17

-showhistory, 17

-usage, 17

Command pane prompt configuration
 option, 232

Command to Window menu item, 69

command window configuration

 option, 214

commands

!, 310, 321

 “ ” (quotation marks-double), 293

Abort, 312

About, 76, 289

AcceptAcString, 275

AddStr, 293

AllowAutoCheckout, 72, 313

AlterLocation (deprecated), 317

AlterMode, 295

 “*any_string*”, 293

AppendTagFile, 27, 70, 307

AutoCompleteList, 275

AutoCompletePrototypeList, 275

AutoIndent, 66, 292

AutoIndentImplicit, 292

AutoIndentOrTab, 292

BackiSearch, 301

Backspace, 309

Beep, 293

 binding, 274

BrowseObjXRef, 64, 315

BugReport, 289

CheckIn, 71, 313

CheckOut, 71, 313

ClearConfig, 74, 282

Close, 58, 285

CloseDependentWindows, 64, 315

CmdPrompt2Wnd (deprecated), 317

Column, 63, 296

CommandToWindow, 69, 310, 321

CommentBlock, 66, 277

Index

ConfigOptions, 73, 282
configure, 171, 282
ConnectByName, 187
ContinueSelection, 302 to 304
Copy, 60, 279
CreateLog (deprecated), 317
CustomizeMenus, 73, 282
Cut, 59, 280
CyclePush, 63, 315
CyclePushBack, 63, 316
Date, 68, 293
debugbutton, 183
DebugByName, 187
Delete, 60
DelUnconfirmedAcString, 275
DiffFiles, 70, 310
Discard, 72, 313
Done, 285
DosFormat, 285
Down, 296
DownSome, 296
DrawWrapLine, 316
editbutton, 183
EditLine, 297
Editor, 274
EditorFlags, 62, 285
EditOtherByName, 187
EditTag (deprecated), 317
EnterInsertMode, 295
EOF, 297
EOL, 297
ErrorOrTag, 307
ExecuteCmd, 69, 310
ExecuteCommandOnSelected, 188
ExecuteShellCommandOnSelected, 188
FileProperties, 58, 285
FlashCursor, 297
GenerateXrefInfo, 64, 316
GetSelectedString, 304
GetSelection, 304
Goto, 297
GotoObjDecl, 64, 316
GotoObjDef, 63, 316
Grep, 68, 310
Help, 75, 289
help regarding, 6
helpview, 6
Identify, 75, 289
identifying for keys or mouse
 clicks, 289
if, 309
Indent, 66, 293
InsertFile, 67, 293
InsertNewline, 294
iSearch, 301
issuing manually, 274
JoinLines, 67, 277
LanguageOptions, 62, 286
Left, 297
LeftSome, 298
LeftU, 298
LineD, 298
LoadConfigFromFile, 74, 282
LoadFile, 286
LoadFileWithNewEditor, 56, 286
LoadModuleByName, 188
LowerCaseBlock, 67, 277
Make, 68, 311
MergeFiles, 70, 311
Minibuffer, 70, 311
MoveCursor, 298
MoveToEndOfSelection..., 276
MultiBar, 68, 311
NewTag, 307
NextAutoCompleteString, 276
NextWindow, 317
NoSelection, 280, 304
Notepad, 69, 311
OpenFile, 56, 286
OpenProjectByName, 188
OpenTag, 308
OpenText (deprecated), 317

Index

PageDown, 298
PageSetup, 57, 286
PageUp, 298
Paste, 60, 281
PlaceUnderVC, 72, 313
PrevAutoCompleteString, 276
PreventAutoCheckout, 72, 313
Print, 58, 286
Project Manager, 187
QuerySaveAll, 57, 287
QuerySaveCheckinAll, 71, 314
QuerySaveComments, 287
Quit, 58, 287
quotation marks-double (“ ”), 293
Quote, 296
RectCopy1, 67, 281
RectCut1, 67, 281
RectPaste1, 67, 281
Redo, 59, 312
RegenerateXrefInfo, 64, 317
RepeatLast, 59, 312
ResetTags, 70, 308
Return, 298
ReverseWord, 299
Revert, 57, 287
RevertDate, 73, 314
RevertHistory, 73, 314
RevertToBackup, 57, 314
RevertVersion, 73, 314
Right, 299
RightD, 299
RightSome, 299
Save, 56, 287
SaveAll, 287
SaveAs, 57, 287
SaveConfig, 74, 283
SaveConfigToFile, 74, 283
Search, 60, 301
SecondarySelectAll, 304
SecondarySelectionAdjust, 305
SecondarySelectionExtend, 305
SecondarySelectionReplace, 305
SecondarySelectionReplaceClip, 305
SecondarySelectionStart, 305
SecondarySelectLine, 304
SecondarySelectWord, 304
SelectAll, 60, 305
SelectionAdjust, 305
SelectionDrop, 284
SelectionExtend, 305
SelectionGrab, 305
SelectionStart, 305
SelectionStartDrag, 284
SelectionStartDragAdd, 284
SelectLanguage, 62, 288
SelectLine, 67, 305
SelectMatch, 305
SelectRange, 306
SelectToLines, 306, 309
SelectToMatch, 63, 306
SelectWord, 306
Shell, 311, 321
ShowContextMenu, 312
ShowHistory, 72, 314
ShowLastEdit, 72, 315
ShowView, 315
SOF, 299
SOL, 299
SOL0, 300
SOL1, 300
SOLSecondary, 306
SpecialTag, 308
StopSearch, 301
strings of, 311
Tab, 294
ToggleErrorView, 296
ToggleReadOnly, 63, 288
ToggleWritePermission, 57, 288
TruncateSearch, 301
UnCommentBlock, 66, 278
Undo, 59, 312
Unindent, 66, 293

Index

- Up**, 300
- UpperCaseBlock**, 66, 278
- UpSome**, 300
- UserName**, 294
- VCBuffer**, 315
- WGUtills**, 70, 311
- Word**, 300
- CommandToWindow** command, 69, 310, 321
- Comment** menu item, 22, 66
- CommentBlock** command, 66, 277
- comments, 38
 - color configuration, 268
 - inserting in code, 38
- Comments stick flush left** configuration option, 255
- Commit**
 - menu item, 71
- Commit Changes** dialog box, 152
 - menus, 153
- comparing files, 45
- Compatibility** configuration option, 219
- Config** menu, 46, 73
- ConfigOptions** command, 73, 282
- configuration
 - alternative editor, 213
 - commands, *see* configuration options
 - options, 206
 - See also* configuration options
- configuration (.cfg) files, 172, 175
 - format, 177
 - loading, 178
 - user, 172
- configuration file
 - global, 175
 - user, 176
- configuration options
 - Ada paren mode**, 256
 - adaContinuationSize**, 258
 - adaindentSize**, 258
 - aiAdaParenindentMode**, 256
 - aiCharsLikeStarinComment**, 255
 - aiCommentsStayFlushLeft**, 255
 - aiimplicitindent**, 253
 - aiimplicitindentinComments**, 254
 - aiimplicitOnlyAtinitial**, 254
 - aiParenindentMode**, 256
 - aiSwitchinTwo**, 254
 - aiTouchComments**, 254
 - Allow beeping**, 209
 - allowExecutionin...** (deprecated), 270
 - allowExecutioninBpCommand**, 241
 - allowMiddleClick**, 252
 - allowOvertypemode**, 253
 - allowProcCallinOsaTask**, 241
 - alwaysUseMeToFixBuildErrors**, 227
 - Application**, 220
 - Ask before halting to set breakpoint**, 230
 - assembly color**, 267
 - assertionSyntaxChecking** (deprecated), 270
 - attemptToShowOldVersion...**, 241
 - Auto indent as you type**, 253
 - Auto indent comments as you type**, 254
 - autoConnectionMode**, 229
 - autoDwarf2Dbo**, 237
 - autoEditErrors**, 226
 - autoEditWarnings**, 227
 - autoGrabHeadFiles**, 259
 - Automatic Checkout**, 215
 - Automatically dereference pointers**, 234
 - Automatically open editor on errors**, 226
 - Automatically save “User Methods” changes**, 263
 - Automatically save changed connection files**, 262
 - Automatically share target connections**, 229

Index

- autoSaveConnectionsInFiles**, 262
- autoSaveUserConnections**, 263
- autoStabs2Dbo**, 237
- autoVerifyRomSections**, 236
- background** *color*, 266
- backgroundMode**, 231
- bDotColor** *color*, 267
- beep**, 209
- blinkingCursor**, 221
- blockRun**, 242
- blockStep**, 235
- bpSyntaxChecking**, 234
- Break Dot** *color*, 267
- breakColor** *color*, 267
- builderPosition** (deprecated), 270
- buildFilesDir**, 264
- buildFileSet**, 264
- buildType**, 227
- C chars aligned like '*' in comments**, 255
- C paren indent mode**, 256
- character** *color*, 268
- Check syntax of breakpoints when they are set**, 234
- clearButtons**, 242
- clearEditButtons**, 259
- clearing**, 179
- clearKeys**, 221
- clearMenus**, 221
- clearMice**, 221
- clickPause**, 221
- clipManLaunch**, 209
- closeButtonOnTitlebar**, 208
- Color C++ comments in C**, 268
- Coloring for multiple debuggers**, 231
- colors**, 265
- colorSyntax**, 267
- Command line arguments**, 214
- Command pane prompt**, 232
- command window**, 214
- comment** *color*, 268
- Comments stick flush left**, 255
- Compatibility**, 219
- Configure Debugger Buttons**, 232
- Configure Editor**, 210
- Configure Editor Buttons**, 253
- Configure Version Control**, 211, 215
- configureFile**, 221
- Connected targets foreground**
 - color*, 269
- containerSizeIncrement**, 240
- Context Arrow** *color*, 267
- Continue running script files on error**, 234
- continuePlaybackFileOnError**, 234
- Control Area** *color*, 266
- cppCommentsInC**, 268
- Create backup files when saving**, 252
- Criteria to decide if a task is in a group**, 239
- cTextSize**, 242
- Ctrl+cursor jump size**, 253
- cursor blinking**, 221
- customized** *color*, 268
- deadCode** *color*, 268
- Debug server timeout in seconds**, 238
- debugButton**, 242
- debugButtonsLoc** (deprecated), 270
- Debugger**, 230
- Debugger child windows**, 232
- Debugger Colors**, 267
- debugServersDir**, 264
- debugServerSet**, 264
- defaultNpwDir**, 227
- deleteDeadTaskFromGroup**, 236
- derefPointer**, 234
- diffHighlight** *color*, 269
- Diffview Colors**, 269
- Directory memory**, 263
- Disable activation follows focus**, 219
- Disable all grouping**, 219
- Disable all window management**, 219

Index

- Disable geometry caching, 219
- Disable grouping in KDE, 219
- disAsmStyle, 222
- Display all numbers/characters as
 - hex, 230
- Display close (x) buttons, 208
- Display typedef type instead of basic type, 235
- displayConnectionType..., 229
- Do Not Color, 231
- Docking Distance, 219
- Docks to, 220
- Does not dock to, 220
- dontUseVC (deprecated), 270
- Double indent size when indenting
 - body of switch, 254
- downloadWindow, 242
- dragAndDrop, 252
- drawWrapLine, 259
- echoCommandsFrom..., 242
- editButton, 260
- editHeight, 257
- editincrFrequency, 258
- editindent, 253
- Editor, 210
- editorBackups, 252
- editParenMatch, 260
- editPrint2Column, 257
- editSomeSize, 253
- editWidth, 257
- Emacs Editor, 210
- Enter spaces in place of tabs, 252
- Escape restores view after
 - iSearch, 209
- escHalts, 236
- exactCase, 209
- execFilesDir, 264
- execFileSet, 264
- Executables/Binaries:, 264
- Execute tools at low priority
 - (-nice), 228
- extEditor_Choice, 210
- extEditor_EmacsArgumentFormat, 214
- extEditor_EmacsPath, 214
- extEditor_EmacsUseCmd, 214
- extEditor_EmacsUseXTerm, 214
- extEditor_OtherArgumentFormat, 214
- extEditor_OtherPath, 214
- extEditor_OtherUseCmd, 214
- extEditor_OtherUseXTerm, 214
- extEditor_ViArgumentFormat, 214
- extEditor_ViPath, 214
- extEditor_ViUseCmd, 214
- extEditor_ViUseXTerm, 214
- File line #, 267
- file relative line numbers, 231
- fileRelLineBg, 267
- fillParagraphColumn, 259
- firstPosition, 233
- focusOnRaise, 222
- font, 212
- foreground *color*, 266
- formatStringMaxDepth, 243
- formatStringMaxLength, 243
- Generate auto-recover file every ...
 - seconds, 258
- genFilesDir, 264
- genFileSet, 264
- geometry, 243
- Global Colors, 266
- globalHeading, 243
- gotoHitsBpAtTargetAddress, 243
- grabTimeout, 222
- GUI Font, 212
- guiFont, 212
- help regarding, 6
- hexMode, 230
- history, 244
- Horizontal scroll bar
 - (deprecated), 270
- hoverValues, 231
- iconGeometry, 244

Index

- iconifiedButtons** (deprecated), 270
- iconify**, 223
- Ignore/Run Away**, 238
- ignoreMotion**, 223
- implicitEvalEcho**, 244
- Increment to maximum container size**, 240
- Indent comments when indenting multiple lines**, 254
- Indent size**, 253
- Initial height in characters**, 257
- Initial position (XxY)**, 233
- Initial width in characters**, 257
- interleavedOutput**, 244
- Interval to refresh Task Manager**, 238
- iSearchReturn**, 209
- kanjiFont**, 213
- keybind**, 191, 213
- Keys**, 213
- keyword** *color*, 268
- Launch clipboard manager**, 209
- leaveTypeDef**, 235
- lineNumberMode**, 231
- linesNonOverlapped**, 244
- Load Color Scheme**, 266
- Location of VC binary**, 215
- longjmpStepMode**, 238
- Match exact case in searches**, 209
- maxContainerDisplaySize**, 240
- Maximize Step Speed**, 238
- Maximum initial size (WxH)**, 233
- maxViewSize**, 233
- maxWindowSize** (deprecated), 270
- menu** command, 189
- menuDelay**, 223
- Menus**, 213
- Minimize Temp Stops**, 238
- Minimum initial size (WxH)**, 233
- minViewSize**, 233
- minWindowSize** (deprecated), 270
- moon**, 211
- More Color Options**, 268
- More Debugger Options**, 232
- More Editor Options**, 253
- Mouse**, 213
- mouse** command, 191
- MULTI Editor**, 210
- multiiconPreName**, 224
- multiWinPreName**, 224
- No Number**, 231
- noDecoration**, 224
- number** *color*, 268
- Number of parallel processes**, 228
- numberOfParallelProcesses**, 228
- numberSeparator**, 224
- On warnings**, 227
- Only auto indent line if first char typed is # or ***, 254
- Open build details window**, 226
- openFilesinNewBuffers**, 252
- osaSwitchToUserTaskAutomatically**, 245
- osaTaskAutoAttachLimit**, 245
- other editor**, 210, 213
- overwriteScriptBreakpoints**, 261
- paddedHex**, 245
- Parallel Build**, 227
- Path to Editor**, 214
- Per File Settings Defaults**, 258
- per session**, 260
- Phase of moon in scroll bar box**, 211
- pointerColor** *color*, 267
- prepareAllCores**, 246
- Pressing Esc halts the target when connected**, 236
- Print 2 columns if landscape orientation is selected**, 257
- printCommand**, 210
- Proc line #**, 267
- procedure relative line numbers**, 231
- procQualifiedLocalimplies...**, 246
- procRelativeLines**, 230

Index

- procRelLineBg**, 267
- Project directory root**, 227
- Project Manager**, 226
- prompt**, 232
- PromptProgramChange**, 239
- promptQuitDebugger**, 239
- quietTogCmd**, 246
- recordCommented...**, 246
- Remember base addresses (e.g. _TEXT)**, 262
- Remember last connect command used for process**, 262
- rememberBaseAddr**s, 262
- rememberBreakpoints**, 261
- rememberDirs**, 263
- rememberWindowPositions**, 208
- requestOsaPackage**, 236
- Restored breakpoints overwrite breakpoints set by scripts**, 261
- Reuse Data Explorer**, 237
- Reuse editor windows**, 252
- Root of checkout**, 216
- runRcScripts**, 247
- “s” (step) and “n” (next) are blocking by default**, 235
- Save arguments between sessions**, 261
- Save data explorers between sessions**, 260
- Save window positions and sizes**, 208
- saveCommandHistory**, 260
- saveDebuggerWindowPos**, 261
- saveDebugServer**, 262
- saveRunArguments**, 261
- saveViewWindows**, 260
- saving changes to**, 172
- Scroll bar width in pixels (deprecated)**, 270
- scrollBarWidth (deprecated)**, 270
- scrollHLocation (deprecated)**, 270
- scrollLocation (deprecated)**, 270
- Selection color**, 266
- selectionMarginWidth**, 258
- serverPollinterval**, 238
- serverTimeout**, 238
- setBpAtAdrinitWhenExe**cing, 247
- setting**, 170
 - with **configure** command, 171
 - with **Options** window, 170
- sharedSymbols**, 247
- shellConfirm**, 247
- Show locations of variables for “print” command**, 235
- Show tool commands (-commands)**, 228
- Show tooltips**, 210
- Show variable values in tooltips**, 231
- showAddress**, 235
- showGrepCommand**, 224
- showPosinNoDisplayMode**, 235
- showProgress**, 226
- showVersionControl**, 265
- silentlyReloadSymbols**, 247
- Single-Thread Build**, 227
- Source Code Font**, 212
- sourceFilesDir**, 264
- sourceFileSet**, 264
- SourceSafe database location**, 216
- Spaces per indent for Ada**, 258
- startedConnectionFg color**, 269
- Status message color**, 267
- Stepping over longjumps**, 238
- stepToBpignores...**, 248
- string color**, 268
- synchronous**, 225
- syntax color settings**, 268
- Syntax Coloring**, 267
- tabsAreSpaces**, 252
- tabSize**, 253
- targetWindowSwitchViewOnBpHit**, 248
- Taskbar Organizer**, 216
- taskMatchCriteria**, 239

Index

- tbTypeBg**, 249
- tbTypeFg**, 249
- tempFileDir**, 257
- toolCommands**, 228
- tooltips**, 210
- traceAlwaysOn** (deprecated), 270
- Translate DWARF debugging information**, 237
- Translate stabs debugging information**, 237
- unifyViewWindows**, 237
- Use Color Offsets**, 231
- Use command window**, 214
- Use icons for buttons** (deprecated), 270
- Use lock files (-lockout)**, 228
- Use MULTI Editor on errors**, 227
- Use Preset Colors**, 231
- Use procedure relative line number (vs. file relative)**, 230
- Use XTerm**, 214
- useFileRelLineBg**, 267
- useLockFiles**, 228
- useLowPriority**, 228
- useProcRelLineBg**, 267
- User Directories**, 264
- useWmPositioning**, 225
- verifyHalt**, 230
- versCtrl_ClearCaseAuto...**, 215
- versCtrl_ClearCasePath**, 215
- versCtrl_CustomAutoCheckout**, 215
- versCtrl_CvsPath**, 215
- versCtrl_PvcsAutoCheckout**, 215
- versCtrl_RcsAutoCheckout**, 215
- versCtrl_SourceSafeAuto...**, 215
- versCtrl_SourceSafeDatabase**, 216
- versCtrl_SourceSafePath**, 215
- versCtrl_SourceSafeRoot**, 216
- versCtrl_SubversionPath**, 215
- Version Control**, 211
- versionControlType**, 211
- Vertical scroll bar** (deprecated), 270
- vi Editor**, 210
- viewDef**, 250
- viewsAreChildrenMode**, 232
- viewUnsignedCharAsint**, 231
- warnOnBpReplacement**, 251
- warnOnCmdAdrLine...**, 251
- warpPointer**, 210
- Window**, 220
- window docking**, 218, 220
- Windows**, 213
- wordWrap**, 258
- wrapColumn**, 258
- wrapindent**, 259
- XTerm**, 214
- configure**
 - at startup, 178
 - buttons, 242
 - colors, 269
 - command, 171, 282
 - during a session, 178
 - file extensions, 202
 - indents, 37
 - MULTI, 73, 170, 172
 - See also* configuration options
 - the MULTI Editor, 46
- Configure Debugger Buttons**
 - configuration option, 232
- Configure Editor Buttons** configuration option, 253
- Configure Editor** configuration option, 210, 213
- Configure Version Control** configuration option, 211, 215
- configureFile** configuration option, 221
- ConnectByName** command, 187
- Connected targets** configuration option, 269
- connection chooser**, display connection type, 229
- connection method**, save, 262

Index

- containerSizeIncrement**
 - configuration option, 240
- Context Arrow** configuration option, 267
- context-sensitive help, 6
- Continue running script files on error**
 - configuration option, 234
- continuePlaybackFileOnError**
 - configuration option, 234
- ContinueSelection** command, 302 to 304
- controlColor** *color* configuration
 - option, 266
- conventions
 - typographical, xii
- copy
 - column of text, 39
 - keyboard shortcuts, 328
- Copy**
 - command, 60, 279
 - key (UNIX), 279, 329
 - menu item, 21, 60
- Copy1, Copy2, Copy3, Copy4**
 - commands, 279
- cppCommentsinC** configuration
 - option, 268
- Create backup files when saving**
 - configuration option, 252
- Create Checkout**, 150
- CreateLog** command (deprecated), 317
- Criteria to decide if a task is in a group**
 - configuration option, 239
- cross reference, 23
- cTextSize** configuration option, 242
- Ctrl key, 284
- Ctrl+– keyboard shortcut, 277
- Ctrl+* keyboard shortcut, 66, 162, 277
- Ctrl+. keyboard shortcut, 59, 312, 335
- Ctrl+' keyboard shortcut, 334
- Ctrl++ keyboard shortcut, 66
- Ctrl+– keyboard shortcut, 67
- Ctrl+/ keyboard shortcut, 333
- Ctrl+; keyboard shortcut, 36
- Ctrl+[keyboard shortcut, 333
- Ctrl+\ keyboard shortcut, 296
- Ctrl+] keyboard shortcut, 333
- Ctrl+^ keyboard shortcut, 325
- Ctrl++ keyboard shortcut, 278
- Ctrl+\ keyboard shortcut, 335
- Ctrl+0 (zero) keyboard shortcut, 300
- Ctrl+cursor jump size** configuration
 - option, 253
- Ctrl+0 (zero) keyboard shortcut, 325
- Ctrl+2 keyboard shortcut, 36, 66, 292
- Ctrl+A keyboard shortcut, 60, 305, 332
- Ctrl+B keyboard shortcut, 28, 301, 332
- Ctrl+Backspace keyboard shortcut, 309, 330
- Ctrl+C keyboard shortcut, 60, 279, 329
- Ctrl+Copy keyboard shortcut (UNIX), 279, 329
- Ctrl+Cut keyboard shortcut (UNIX), 280, 328
- Ctrl+D keyboard shortcut, 60, 309, 330
- Ctrl+Delete keyboard shortcut, 309, 330
- Ctrl+DownArrow keyboard shortcut, 296, 324
- Ctrl+E keyboard shortcut, 297, 326
- Ctrl+End keyboard shortcut, 297, 326
- Ctrl+Enter keyboard shortcut, 326
- Ctrl+F keyboard shortcut, 28, 60, 301, 332
- Ctrl+G keyboard shortcut, 26, 61, 297, 326
- Ctrl+H keyboard shortcut, 298, 325
- Ctrl+Home keyboard shortcut, 299, 325
- Ctrl+i keyboard shortcut, 66, 293, 334
- Ctrl+J keyboard shortcut, 296, 324
- Ctrl+K keyboard shortcut, 300, 324
- Ctrl+L keyboard shortcut, 299, 325
- Ctrl+LeftArrow keyboard shortcut, 299, 325
- Ctrl+Left-click, 284, 336
- Ctrl+M keyboard shortcut, 67, 309, 331

Index

- Ctrl+N keyboard shortcut, 56, 286, 327
- Ctrl+O keyboard shortcut, 56, 162, 286, 327
- Ctrl+P keyboard shortcut, 67, 277, 331
- Ctrl+Paste keyboard shortcut (UNIX), 281, 330
- Ctrl+Q keyboard shortcut, 58, 162, 285, 327
- Ctrl+RightArrow keyboard shortcut, 300, 325
- Ctrl+[Right-click] mouse action, 338
- Ctrl+S keyboard shortcut, 56, 287, 327
- Ctrl+Shift+Tab keyboard shortcut, 316
- Ctrl+T keyboard shortcut, 307, 336
- Ctrl+Tab keyboard shortcut, 63, 315, 326
- Ctrl+U keyboard shortcut, 309, 330
- Ctrl+UpArrow keyboard shortcut, 300, 324
- Ctrl+V keyboard shortcut, 60, 281, 329
- Ctrl+W keyboard shortcut, 299, 325
- Ctrl+X keyboard shortcut, 59, 280, 328
- Ctrl+Y keyboard shortcut, 59, 312, 328
- Ctrl+Z keyboard shortcut, 59, 312, 328
- Ctrl+Shift+A keyboard shortcut, 335
- Ctrl+Shift+B keyboard shortcut, 298, 324, 333
- Ctrl+Shift+C keyboard shortcut, 279, 329
- Ctrl+Shift+Copy keyboard shortcut (UNIX), 279, 329
- Ctrl+Shift+Cut keyboard shortcut (UNIX), 280, 329
- Ctrl+Shift+DownArrow keyboard shortcut, 302, 331
- Ctrl+Shift+E keyboard shortcut, 303
- Ctrl+Shift+Enter keyboard shortcut, 303, 332
- Ctrl+Shift+F keyboard shortcut, 29, 60, 301
- Ctrl+Shift+G keyboard shortcut, 25, 61, 297
- Ctrl+Shift+H keyboard shortcut, 314, 335
- Ctrl+Shift+i keyboard shortcut, 66, 293, 334
- Ctrl+Shift+L keyboard shortcut, 315, 335
- Ctrl+Shift+LeftArrow keyboard shortcut, 303, 332
- Ctrl+Shift+N keyboard shortcut, 298, 325
- Ctrl+Shift+O, 162, 286, 327
- Ctrl+Shift+P keyboard shortcut, 58
- Ctrl+Shift+PageDown keyboard shortcut, 303, 331
- Ctrl+Shift+Paste keyboard shortcut (UNIX), 281
- Ctrl+Shift+Q keyboard shortcut, 285, 327
- Ctrl+Shift+R keyboard shortcut, 294, 335
- Ctrl+Shift+RightArrow keyboard shortcut, 304, 332
- Ctrl+Shift+[Right-click] mouse action, 338
- Ctrl+Shift+S keyboard shortcut, 57, 287, 327
- Ctrl+Shift+T keyboard shortcut, 280, 335
- Ctrl+Shift+Tab keyboard shortcut, 63, 326
- Ctrl+Shift+U keyboard shortcut, 66, 278
- Ctrl+Shift+UpArrow keyboard shortcut, 304, 331
- Ctrl+Shift+V keyboard shortcut, 281, 329
- Ctrl+Shift+X keyboard shortcut, 280, 328
- curly braces {}, matching, 63
- cursor movement
 - beginning of next line, 298
 - columns, 63

Index

- flashing to current line, 63
- keyboard shortcuts, 324
- when dialog box opens, 210
- cursor position, 22
- customize
 - GUI, 181
 - icons, 185
 - keys, 191
 - menus, 186
 - mouse clicks, 191
 - toolbar buttons, 183
- Customize Menus** menu item, 73
- customized** *color* configuration
 - option, 268
- CustomizeMenus** command, 73, 282
- cut
 - column of text, 40
 - keyboard shortcuts, 328
- Cut**
 - command, 59, 280
 - key (UNIX), 280, 328
 - menu item, 21, 59
- Cut Lines** menu item, 67
- Cut1, Cut2, Cut3, Cut4** commands, 280
- CVS Checkout Editor**
 - menu item, 146
 - using, 149
- CVS version control
 - configuration, 123, 211
 - features, 149
 - integration, 129, 149
 - path, 215
 - `_cwd` Launcher variable, 116
- CyclePush** command, 63, 315
- CyclePushBack** command, 63, 316
- D**
- Data Explorer
 - nested structs, 243
 - position, 233
 - save between sessions, 260
 - value length, 243
 - window format, 250
 - window overlap, 244
 - window size, 233
- database location, SourceSafe, 216
- Date** command, 68, 293
- deadCode** *color* configuration option, 268
- Debug server timeout in seconds**
 - configuration option, 238
- Debug Servers** configuration option, 264
- debugbutton** command, 183
- debugButton** configuration option, 242
- debugButtonsLoc** configuration option
 - (deprecated), 270
- DebugByName** command, 187
- Debugger
 - colors, 267
 - configuration options, 230
 - history, 244
 - panes, 244
 - save window position, 261
 - tab, 230
 - window size and position, 243
- Debugger child windows** configuration
 - option, 232
- Debugger Colors** configuration
 - option, 267
- Debugger commands
 - > (minus greater than) menu
 - command, 190
 - configureFile**, 221
 - keybind**, 213
 - keybind** command, 191
 - mouse** command, 191
- debugging_window* menu item, 74
- debugServersDir** configuration
 - option, 264
- debugServerSet** configuration option, 264
- defaultNpwDir** configuration option, 227
- define new language, 47

Index

Delete

- command, 60
- key, 309, 330
- keyboard shortcuts, 330
- menu item, 60

Delete dead tasks from group

- configuration option, 236

deleting text, 60

DelUnconfirmedAcString command, 275

derefPointer configuration option, 234

detail pane, MULTI Launcher, 97

dialog boxes

- cursor movement in, 210

-diff command line option, 17

Diff Selected Files menu item, 153

Diff Viewer

- opening, 157
- overview, 157
- starting from the command line, 158

DiffFiles

- command, 70, 310
- menu item, 45, 70, 157

diffHighlight *color* configuration option, 269

Diffview Colors configuration option, 269

Directory memory configuration option, 263

Disable activation follows focus configuration option, 219

Disable all grouping configuration option, 219

Disable all window management configuration option, 219

Disable geometry caching configuration option, 219

Disable grouping in KDE configuration option, 219

disable version control, 122, 211

disAsmStyle configuration option, 222

Discard Changes menu item, 72

Discard command, 72, 313

Display all numbers/characters as hex configuration option, 230

Display close (x) buttons configuration option, 208

Display connection type in Connection Chooser configuration option, 229

Display typedef type instead of basic type configuration option, 235

Do Not Color configuration option, 231

Docking Distance configuration option, 219

docking, window, *see* window docking

Docks to configuration option, 220

document set, x to xi

Does not dock to configuration option, 220

Done command, 285

dontUseVC configuration option (deprecated), 270

DOS Format

- menu item, 61

DosFormat command, 285

Double indent size when indenting body of switch configuration option, 254

[Double Left-click] mouse action, 337

[Double Middle-click] mouse action, 337

Down command, 296

DownArrow key, 296, 324

downloadWindow configuration option, 242

DownSome command, 296
size of, 253

Drag and drop text editing configuration option, 252

drawWrapLine configuration option, 259

DrawWrapLine command, 316

Index

E

echoCommandsFrom... configuration option, 242

Edit menu, 59

Edit Selected Files menu item, 153

editbutton

command, 183

editButton

configuration option, 260

editHeight configuration option, 257

editincrFrequency configuration

option, 258

editindent configuration option, 253

EditLine command, 297

Editor

auto recover, 258

commands, 274

See also commands

configuration option, 210

main window, picture of, 19

margin width, 258

merging three files, 43

merging two files, 42

progress window, 112

specifying an alternate, 320

Toolbar, 20

window size, 257

Editor Help menu item, 75

Editor menus

Block menu, 66

Config menu, 73

Edit menu, 59

File menu, 56

Help menu, 75

Tools menu, 68

Version menu, 71

View menu, 62

Windows menu, 74

editorBackups configuration option, 252

EditorFlags command, 62, 285

EditOtherByName command, 187

editParenMatch configuration

option, 260

editPrint2Column configuration

option, 257

editSomeSize configuration option, 253

EditTag command (deprecated), 317

editWidth configuration option, 257

Either radio button, 30

Emacs Editor

command line arguments, 214

configuration option, 210

path, 214

use command window, 214

use XTerm, 214

End key, 297, 326

Ends line check box, 31

Ends word check box, 31

Enter key, 294, 336

Enter spaces in place of tabs configuration

option, 252

EnterInsertMode command, 295

EOF command, 297

EOL command, 297

ErrorOrTag command, 307

errors and warnings

always view in MULTI Editor, 227

Esc key, 63, 297, 301, 312, 333, 336

Escape restores view after iSearch

configuration option, 209

escHalts configuration option, 236

exactCase configuration option, 209

execFilesDir configuration option, 264

execFileSet configuration option, 264

Execute Editor Commands

dialog box, 274, 311

menu item, 70

Execute Shell Command menu item, 69

Execute tools at low priority (-nice)

configuration option, 228

ExecuteCmd command, 69, 310

Index

ExecuteCommandOnSelected

command, 188

ExecuteShellCommandOnSelected

command, 188

Exit All menu item, 58

extEditor_Choice configuration

option, 210

extEditor_EmacsArgumentFormat

configuration option, 214

extEditor_EmacsPath configuration

option, 214

extEditor_EmacsUseCmd configuration

option, 214

extEditor_EmacsUseXTerm

configuration option, 214

extEditor_OtherArgumentFormat

configuration option, 214

extEditor_OtherPath configuration

option, 214

extEditor_OtherUseCmd configuration

option, 214

extEditor_OtherUseXTerm configuration

option, 214

extEditor_ViArgumentFormat

configuration option, 214

extEditor_ViPath configuration

option, 214

extEditor_ViUseCmd configuration

option, 214

extEditor_ViUseXTerm configuration

option, 214

F

F1 key, 75, 289

F1 key, for help, 6

file chooser

directory memory, 263

showing version control

information, 265

user directories, 264

file chooser dialog box (UNIX), 76

file command line option, 17

file extensions, 48

configuring, 202

File line # configuration option, 267

File menu, 56

File Number configuration option, 231

FileProperties

command, 58, 285

fileRelLineBg configuration option, 267

files

checking in, 71

checking in all, 71

checking out, 71

checkout automatically, 72

color in browser, 249

comparing, 45

differences between, 310

discarding changes, 72

inserting, 67

keyboard shortcuts, 327

merging, 40

merging three into one, 43

merging two into one, 42

opening from the Debugger, 18

opening from the Project Manager, 18

switching read/write modes, 57, 63,
288

version control, 72

Fill Paragraph Column configuration

option, 259

Find

button, 29 to 30

key (UNIX), 301, 333

menu item, 29, 60

Find then Replace button, 30

first match auto-completion, 53

firstPosition configuration option, 233

FlashCursor

command, 297

menu item, 63

floating point number form, 52

Index

focusOnRaise configuration option, 222
font configuration option, 212
foreground *color* configuration option, 266
formatStringMaxDepth configuration option, 243
formatStringMaxLength configuration option, 243
Forward radio button, 30
Full Rescan, 140
function prototypes, 23
 auto-complete, 53
 automatically grab, 62, 259
 display, 54
 show, 62
functions, color in browser, 249

G

gbugrpt utility, 289
General Files configuration option, 264
Generate auto-recover file every ...
 seconds configuration option, 258
Generate Cross References
 menu item, 22, 25, 64
 overview, 25
GenerateXrefInfo command, 64, 316
genFilesDir configuration option, 264
genFileSet configuration option, 264
geometry configuration option, 243
GetSelectedString command, 304
GetSelection command, 304
global
 configuration file, 175
 script file, 180
Global Colors configuration option, 266
global configuration file, 175
 override, 176
global Launcher variables, 116
Global Options tab, 218 to 219
global script file, 180
globalHeading configuration option, 243
Go To Declaration menu item, 22, 24
Go To Declaration of Selected Object
 menu item, 64
Go To Definition menu item, 22, 24
Go To Definition of Selected Object
 menu item, 63
Goto
 line number, 26
 menu item, 25
GoTo
 command, 297
 dialog box, 25
 menu item, 61
gotoHitsBpAtTargetAddress
 configuration option, 243
GotoObjDecl
 command, 64
GotoObjDecl command, 316
GotoObjDef
 command, 63
GotoObjDef command, 316
grab function prototypes, 53
grabTimeout configuration option, 222
grep
 command, 33, 68, 310
 dialog box, *see* Search in Files
 licensing, 34
 print full command, 224
 utility, 68
grouping, window, 199
 See also window docking
GUI Font configuration option, 212
guiFont configuration option, 212

H

-h command line option, 17
help, *see* online help
Help
 command, 75, 289

Index

- menu, 75
- menu in Editor, 75
- help** command line option, 17
- Help** menu
 - online help, obtaining, 6
- Help Viewer
 - navigating, 8 to 10
 - opening additional manuals in, 11
 - window, 7
- helpview** command, 6
- hex integers, language support, 51
- hex values, display, 245
- hexMode** configuration option, 230
- History Browser, 72, 154
 - branch, 155
 - comments, 155
 - current tag, 155
 - menu, 156
 - opening, 154
 - shortcut menu, 156
 - status, 155
- History Browser menu items
 - Checkout Browser**, 156
 - Close**, 156
 - Diff 2 Versions**, 156
 - Diff with Local Version**, 156
 - Edit Local Version**, 156
 - Refresh History**, 156
 - Revert to**, 157
 - Revert to Selected Version**, 156
 - View**, 157
 - View Selected Version**, 156
- history** configuration option, 244
- Home key, 300, 325
- Horizontal scroll bar** configuration option (deprecated), 270
- host environment Launcher variables, 117
- hoverValues** configuration option, 231
- I**
- iconGeometry** configuration option, 244
- iconifiedButtons** configuration option (deprecated), 270
- iconify** configuration option, 223
- icons, 20
 - customizing, 185
 - naming, 224
 - size in Debugger, 244
- IDE_tool** menu item, 75
- Identify**
 - command, 75, 289
 - menu item, 75
- identifying boundaries of a block, 38
- if** command, 290, 309
- if** commands, 309
- Ignore/Run Away** configuration option, 238
- ignoreMotion** configuration option, 223
- implicitEvalEcho** configuration option, 244
- Import Commit Log** menu item, 153
- Increment to maximum container size** configuration option, 240
- incremental search, 28, 301
- indent
 - characters, 37
 - configure, 37
 - keyboard shortcuts, 334
 - text, 36
 - text, automatically, 36
- Indent**
 - command, 66, 293
 - menu item, 22, 66
- Indent comments when indenting multiple lines**, 254
- Indent size** configuration option, 253
- index.gmb** configuration file, 106, 114
- index.gsc** configuration file, 47
- Initial height in characters** configuration option, 257
- Initial position (XxY)** configuration option, 233

Index

Initial width in characters configuration

option, 257

Insert Date menu item, 68

Insert File menu item, 67

insert mode, 295

InsertFile command, 67, 293

inserting

comments in code, 38

literal characters, 197

InsertNewline command, 294

Integrated Development Environment

(IDE), *see* MULTI Integrated

Development Environment (IDE)

INTEGRITY BSP workspaces,

creating, 103

interleavedOutput configuration

option, 244

Interval to refresh Task Manager

configuration option, 238

iSearch command, 301

iSearchReturn configuration option, 209

J

JoinLines

command, 67, 277

menu item, 67

K

Kanji Font configuration option, 213

key bindings, clear all, 221

key sequences

binding commands, 295

keybind

command, 191, 324

configuration option, 213

keyboard shortcuts

auto-complete, 333

clipboard, 328

copy, cut and paste, 328

Ctrl+–, 67

Ctrl+– (Ctrl + minus), 277

Ctrl+*, 277

Ctrl++, 66

Ctrl+., 312

Ctrl+;, 292

Ctrl+\\, 296

Ctrl++ (Ctrl + plus), 278

Ctrl+*, 66

Ctrl+., 59

Ctrl+2, 66

Ctrl+A, 60

Ctrl+C, 60

Ctrl+D, 60

Ctrl+F, 301

Ctrl+G, 297

Ctrl+i, 293

Ctrl+M, 67

Ctrl+N, 56

Ctrl+O, 286

Ctrl+P, 67

Ctrl+Q, 58

Ctrl+S, 287

Ctrl+Shift+F, 60

Ctrl+Shift+G, 297

Ctrl+Shift+i, 66

Ctrl+Shift+S, 287

Ctrl+Shift+Tab, 316

Ctrl+Shift+U, 278

Ctrl+Tab, 315

Ctrl+V, 60, 281

Ctrl+X, 59

Ctrl+Y, 312

Ctrl+Z, 59

Ctrl+0 (zero), 300

Ctrl+2, 292

Ctrl+A, 305

Ctrl+B, 301

Ctrl+Backspace, 309

Ctrl+C, 279

Ctrl+D, 309

Ctrl+Delete, 309

Index

Ctrl+DownArrow, 296
Ctrl+E, 297
Ctrl+End, 297
Ctrl+F, 60
Ctrl+G, 61
Ctrl+H, 298
Ctrl+Home, 299
Ctrl+i, 66
Ctrl+J, 296
Ctrl+K, 300
Ctrl+L, 299
Ctrl+LeftArrow, 299
Ctrl+M, 309
Ctrl+N, 286
Ctrl+O, 56
Ctrl+P, 277
Ctrl+Q, 285
Ctrl+RightArrow, 300
Ctrl+S, 56
Ctrl+Shift+B, 298
Ctrl+Shift+C, 279
Ctrl+Shift+DownArrow, 302
Ctrl+Shift+E, 303
Ctrl+Shift+Enter, 303
Ctrl+Shift+F, 301
Ctrl+Shift+G, 61
Ctrl+Shift+H, 314
Ctrl+Shift+i, 293
Ctrl+Shift+L, 315
Ctrl+Shift+LeftArrow, 303
Ctrl+Shift+N, 298
Ctrl+Shift+O, 286
Ctrl+Shift+P, 58
Ctrl+Shift+PageDown, 303
Ctrl+Shift+Q, 285
Ctrl+Shift+R, 294
Ctrl+Shift+RightArrow, 304
Ctrl+Shift+S, 57
Ctrl+Shift+T, 280
Ctrl+Shift+Tab, 63
Ctrl+Shift+U, 66
Ctrl+Shift+UpArrow, 304
Ctrl+Shift+V, 281
Ctrl+Shift+X, 280
Ctrl+T, 307
Ctrl+Tab, 63
Ctrl+U, 309
Ctrl+UpArrow, 300
Ctrl+V, 281
Ctrl+W, 299
Ctrl+X, 280
Ctrl+Y, 59
Ctrl+Z, 312
cursor movement, 324
default, 324
delete, 330
File commands, 327
indent, 334
miscellaneous, 335
navigating, 324
search, 332
Shift+DownArrow, 302
Shift+End, 303
Shift+Home, 304
Shift+LeftArrow, 303
Shift+PageUp, 303
Shift+RightArrow, 303
Shift+UpArrow, 304
text selection, 331
undo / redo, 328
version control, 335
keyboard shortcuts (UNIX)
 Again, 312
 Copy, 279
 Ctrl+Copy, 279
 Ctrl+Cut, 280
 Ctrl+L10, 280
 Ctrl+L6, 279
 Ctrl+L8, 281
 Ctrl+Paste, 281
 Ctrl+Shift+Copy, 279
 Ctrl+Shift+Cut, 280

Index

Ctrl+Shift+L10, 280
Ctrl+Shift+L6, 279
Ctrl+Shift+L8, 281
Ctrl+Shift+Paste, 281
Cut, 280
Find, 301
L10, 280
L2, 312
L4, 312
L6, 279
L7, 286
L8, 281
L9, 301
Meta+Ctrl+C, 279
Meta+Ctrl+Shift+C, 279
Meta+Ctrl+Shift+V, 281
Meta+Ctrl+Shift+X, 280
Meta+Ctrl+V, 281
Meta+Ctrl+X, 280
Open, 286
Paste, 281
Shift+Copy, 279
Shift+Cut, 280
Shift+Find, 301
Shift+L10, 280
Shift+L6, 279
Shift+L8, 281
Shift+L9, 301
Shift+Paste, 281
Undo, 312

keys

attaching a menu to, 189
Backspace, 309
customizing, 191
Delete, 309
DownArrow, 296
End, 297
Enter, 294
Esc, 63, 297, 301, 312
F1, 75, 289
Home, 300

identifying command for, 289
LeftArrow, 298
PageDown, 298
PageUp, 298
RightArrow, 299
specifying actions and locations
 for, 191
Tab, 292
UpArrow, 300

Keys configuration option, 213

keyword *color* configuration option, 268

L

language

define new, 47
definition file, 48
setting in the Editor, 47
syntax definition file, 49

Language

menu item, 62
submenu, modifying entries, 48

LanguageOptions command, 62, 286

Launch clipboard manager configuration
option, 209

Launch Utility Programs menu item, 70,
90

Launcher, *see* MULTI Launcher

Launcher menu item, 68

leaveTypeDef configuration option, 235

Left command, 297

[Left-click] + Drag mouse action, 336

[Left-click] mouse action, 336

Left-release mouse action, 284

LeftArrow key, 298, 325

LeftSome command, 298

size of, 253

LeftU command, 298

License Info menu item, 76

+line command line option, 17

line number field, 21

Index

Line numbers in source pane

configuration option, 231

LineD command, 298

lines

cutting, 67

inserting, 294

joining, 67

merging, 67

selecting, 67

linesNonOverlapped configuration

option, 244

Load Color Scheme configuration

option, 266

Load Configuration menu item, 74

LoadConfigFromFile command, 74, 282

LoadFile command, 286

LoadFileWithNewEditor command, 56, 286

LoadModuleByName command, 188

Local Rescan, 140

location fields, 21

Location of VC binary configuration

option, 215

log file, creating, 313

longjmpStepMode configuration

option, 238

LowerCase menu item, 67

LowerCaseBlock command, 67, 277

M

macros, 179

Make

command, 68, 311

menu item, 68

Manuals menu item, 75

Match exact case in searches

configuration option, 209

Match menu item, 63

maxContainerDisplaySize configuration

option, 240

Maximize Step Speed configuration

option, 238

Maximum initial size (WxH)

configuration option, 233

maximum match, 275

Maximum size for container display

configuration option, 240

maxViewSize configuration option, 233

maxWindowSize configuration option (deprecated), 270

menu

command, 189

customization, 186

defining, 189

opening, 190

menu (->) command, 190

menu items

1, 2, 3, 4, 5, 6, 7, 8, 58

About MULTI, 76

Append TagFile, 70

application_name, 74

Auto Checkout, 72

Auto Indent, 66

Bookmarks, 75

Browse References of Selected

Object, 64

Check In, 71

Check In All, 71

Check Out, 71

Clear User Default Configuration, 74

Close Dependent Windows, 64

Close Editor, 58

Close File, 58

Column, 63

Command to Window, 69

Comment, 66

Commit, 71

Copy, 60

Customize Menus, 73

Cut, 59

Cut Lines, 67

Index

- debugging_window*, 74
- Delete**, 60
- Diff Files**, 70
- Discard Changes**, 72
- DOS Format**, 61
- Editor Help**, 75
- Execute Editor Commands**, 70
- Execute Shell Command**, 69
- Exit All**, 58
- File Properties**, 58
- Find**, 60
- Flash Cursor**, 63
- Generate Cross References**, 64
- Go To Declaration of Selected Object**, 64
- Go To Definition of Selected Object**, 63
- Goto**, 61
- IDE_tool*, 75
- Identify**, 75
- Indent**, 66
- Insert Date**, 68
- Insert File**, 67
- Join Lines**, 67
- Language**, 62
- Launch Utility Programs**, 70, 90
- Launcher**, 68
- License Info**, 76
- Load Configuration**, 74
- LowerCase**, 67
- Make**, 68
- Manuals**, 75
- Match**, 63
- Merge Files**, 70
- Misc**, 75
- New Editor**, 56
- Next File**, 63
- Notepad**, 69
- Open**, 56
- Options**, 73
- Page Setup**, 57
- Paste**, 60
- Per File Settings**, 62
- Per Language Settings**, 62
- Place Under VC**, 72
- Previous File**, 63
- Print**, 58
- Read Only Window**, 63
- Rect Copy**, 39, 67
- Rect Cut**, 40, 67
- Rect Paste**, 40, 67
- Redo**, 59
- Regenerate Cross References**, 64
- Repeat Last Edit**, 59
- Reset Tags**, 70
- Revert to Backup**, 57
- Revert To Date**, 73
- Revert To History**, 73
- Revert to Saved**, 57
- Revert To Version**, 73
- Save**, 56
- Save All**, 57
- Save As**, 57
- Save Configuration As**, 74
- Save Configuration as User Default**, 74
- Search in Files**, 68
- Select All**, 60
- Show History**, 72
- Show Last Edit**, 72
- Toggle Write Permission**, 57
- UnComment**, 66
- Undo**, 59
- Unindent**, 66
- Updatel**, 71
- UpperCase**, 66
- menuDelay** configuration option, 223
- Menus**
 - clear all, 221
 - configuration option, 213
- MergeFiles**
 - command, 70, 311

Index

- menu item, 70
- merging files, 40
- Meta + Ctrl + C keyboard shortcut (UNIX), 279
- Meta + Ctrl + Shift + C keyboard shortcut (UNIX), 329
- Meta + Ctrl + Shift + V keyboard shortcut (UNIX), 330
- Meta + Ctrl + Shift + X keyboard shortcut (UNIX), 329
- Meta + Ctrl + V keyboard shortcut (UNIX), 330
- Meta + Ctrl + X keyboard shortcut (UNIX), 328
- Meta + Ctrl + C keyboard shortcut (UNIX), 329
- Meta + Ctrl + Shift + C keyboard shortcut (UNIX), 279
- Meta + Ctrl + Shift + V keyboard shortcut (UNIX), 281
- Meta + Ctrl + Shift + X keyboard shortcut (UNIX), 280
- Meta + Ctrl + V keyboard shortcut (UNIX), 281
- Meta + Ctrl + X keyboard shortcut (UNIX), 280
- [Middle-click] mouse action, 337
- Minibuffer** command, 70, 311
- Minimize Temp Stops** configuration option, 238
- Minimum initial size (WxH)** configuration option, 233
- minimum string length, 53
- minViewSize** configuration option, 233
- minWindowSize** configuration option (deprecated), 270
- Misc** menu item, 75
- Modify Checkout**, 151
- moon** configuration option, 211
- More Color Options** configuration option, 268
- dialog box, 269
- More Debugger Options** button, 234 configuration option, 232 dialog box, 234
- More Editor Options** configuration option, 253 dialog box, 257
- most recently used file list, 21
- mouse
 - attaching a menu to, 189
 - default bindings, 336
 - default functions, 336
 - identify commands for, 289
 - ignoring motion of, 223
- Mouse
 - command, 191
 - configuration option, 213
 - defining functions for, 191
- mouse actions, 336
 - Ctrl + Left-click, 284, 336
 - Ctrl + [Right-click], 338
 - Ctrl + Shift + [Right-click], 338
 - [Double Left-click], 337
 - [Double Middle-click], 337
 - Double-left-click, 306
 - Left-click, 284, 336
 - [Left-click] + Drag, 336
 - Left-release, 284
 - [Middle-click], 337
 - [Quadruple Left-click], 337
 - [Quadruple Middle-click], 337
 - Quadruple-left-click, 305
 - Right-click, 312, 338
 - Shift + Right-click, 63
 - Shift + [Double Right-click], 338
 - Shift + [Left-click], 336
 - Shift + Right-click, 306, 338
 - [Triple Left-click], 337
 - [Triple Middle-click], 337
 - Triple-left-click, 305

Index

- mouse bindings
 - clear all, 221
- mouse clicks
 - customizing, 191
 - time between, 221
- MoveCursor** command, 298
- MoveToEndOfSelection...**
 - command, 276
- MULTI Editor**
 - configuration option, 210
 - tab, 251
 - See also* Editor
- MULTI Integrated Development Environment (IDE)
 - configuring, 170, 172
 - configuring at startup, 178
 - configuring during a session, 178
 - document set, xi
 - online help, 6
 - overview, 2
- MULTI Launcher
 - configuring, 91
 - detail pane, 97
 - GUI, 84
 - managing running actions, 112
 - menus, 85
 - overview, 4, 84
 - shortcuts, editing from, 88
 - toolbar, 94
 - utilities, 90
 - variables, *see* MULTI Launcher variables
 - workspaces, editing from, 88
- MULTI Launcher variables
 - creating, 117
 - editing, 117
 - overview, 115
 - pre-defined, 116
 - referencing, 118
 - substitution, 118
 - types of, 116
- MULTI shortcuts, *see* shortcuts, MULTI
- MULTI workspaces, *see* workspaces
- `_multi_dir` Launcher variable, 117
- `_multi_major_version` Launcher variable, 117
- `_multi_micro_version` Launcher variable, 117
- `_multi_minor_version` Launcher variable, 117
- MultiBar** command, 68, 311
- multiiconPreName** configuration option, 224
- multiWinPreName** configuration option, 224
- N**
- navigating
 - keyboard shortcuts, 324
- navigating files, 23
- new** command line option, 17
- New Editor** menu item, 56
- NewTag** command, 307
- Next File** menu item, 63
- NextAutoCompleteString** command, 276
- NextWindow** command, 317
- No Number** configuration option, 231
- noDecoration** configuration option, 224
- Normal** radio button, 31
- NoSelection** command, 280, 304
- Notepad**
 - command, 69, 311
 - menu item, 69
- number** *color* configuration option, 268
- Number of parallel processes**
 - configuration option, 228
- numberOfParallelProcesses**
 - configuration option, 228
- numberSeparator** configuration option, 224

Index

O

On warnings configuration option, 227

online help, 6

See also Help Viewer

for commands, 6

for configuration options, 6

context-sensitive, 6

limitations, 10

online manuals, 6

overview, 6

Only auto indent line if first char typed

is # or * configuration option, 254

Open

key (UNIX), 286, 327

menu item, 56

Open build details window configuration option, 226

Open New Editor Window menu item, 22

OpenFile command, 56, 286

openFilesinNewBuffers configuration option, 252

opening a file

from the Debugger, 18

from the Project Manager, 18

OpenProjectByName command, 188

OpenTag command, 308

OpenText command (deprecated), 317

Options

configuration, *see* configuration options

menu item, 73

window, 170

OSA

package prompt, 236

osaSwitchToUserTaskAutomatically configuration option, 245

osaTaskAutoAttachLimit configuration option, 245

other editor

command line arguments, 214

configuration option, 210

path, 214

use command window, 214

use XTerm, 214

overtyping mode, 253

overwriteScriptBreakpoints

configuration option, 261

P

paddedHex configuration option, 245

Page Setup menu item, 57

PageDown

command, 298

key, 298, 325

PageSetup command, 57, 286

PageUp

command, 298

key, 298, 324

Parallel Build configuration option, 227

parentheses, pause for matching, 260

paste

column of text, 40

keyboard shortcuts, 328

Paste

command, 60, 281

key (UNIX), 281, 329

menu item, 21, 60

Paste1, Paste2, Paste3, Paste4

commands, 281

Path to Editor configuration option, 214

Per File Settings

dialog box, 64

menu item, 62

Per File Settings Defaults configuration option, 258

Per Language Settings menu item, 62

Phase of moon in scroll bar box

configuration option, 211

PlaceUnderVC command, 72, 313

pointerColor *color* configuration option, 267

pre-defined Launcher variables, 116

Index

- prepareAllCores** configuration
 - option, 246
 - Pressing Esc halts the target when connected** configuration option, 236
 - PrevAutoCompleteString** command, 276
 - PreventAutoCheckout** command, 72, 313
 - Previous File** menu item, 63
 - Print**
 - command, 58, 286
 - menu item, 58
 - Print 2 columns if landscape orientation is selected** configuration option, 257
 - Print command** configuration option, 210
 - Print Setup** dialog box (UNIX), 78
 - printing
 - in UNIX, 78
 - Proc line #** configuration option, 267
 - Proc Number** configuration option, 231
 - procedure drop-down menu, 21
 - procedure relative line numbers
 - configuration option, 230
 - processes
 - managing, 112
 - procQualifiedLocalimplies...**
 - configuration option, 246
 - procRelativeLines** configuration option, 230
 - procRelLineBg** configuration option, 267
 - program
 - script file, 180
 - program script file, 181
 - progress window, 112
 - Project directory root** configuration option, 227
 - Project Files** configuration option, 264
 - Project Manager
 - commands, 187
 - configuration options, 226
 - tab, 226
 - prompt** configuration option, 232
 - Prompt for OSA package when package is not found** configuration option, 236
 - Prompt when exiting Debugger**
 - configuration option, 239
 - Prompt when program changes**
 - configuration option, 239
 - PromptProgramChange** configuration option, 239
 - promptQuitDebugger** configuration option, 239
 - Properties** menu item, 22, 58
 - PVCS version control
 - automatic checkout, 215
 - configuration, 124, 211
 - integration, 130
- ## Q
- [Quadruple Left-click] mouse action, 337
 - [Quadruple Middle-click] mouse action, 337
 - QuerySaveAll** command, 57, 287
 - QuerySaveCheckinAll** command, 71, 314
 - QuerySaveComments** command, 287
 - quietTogCmd** configuration option, 246
 - Quit** command, 58, 287
 - quotation marks-double (“ ”)
 - command, 293
 - Quote** command, 296
- ## R
- radio button, 31
 - RCS version control
 - automatic checkout, 215
 - configuration, 123, 211
 - integration, 131
 - Read Only Window** menu item, 63
 - Read-only window indicator, 22
 - recordCommented...** configuration option, 246

Index

- Rect Copy** menu item, 39, 67
 - Rect Cut** menu item, 40, 67
 - Rect Paste** menu item, 40, 67
 - rectangular text section
 - copying, 67, 281
 - cutting, 67, 281
 - pasting, 67, 281
 - RectCopy1** command, 67, 281
 - RectCut1** command, 67, 281
 - RectPaste1** command, 67, 281
 - Redo**
 - command, 59, 312
 - menu item, 59
 - referencing
 - Launcher variables, 118
 - Regenerate Cross References** menu item, 22, 64
 - RegenerateXrefInfo** command, 64, 317
 - regular expressions, 32
 - in search strings, 31
 - Remember base addresses (e.g. _TEXT)**
 - configuration option, 262
 - Remember breakpoints** configuration option, 261
 - Remember last connect command used for process** configuration option, 262
 - rememberBaseAddr**s configuration option, 262
 - rememberDirs** configuration option, 263
 - rememberWindowPositions**
 - configuration option, 208
 - Remove Selected Files** menu item, 153
 - Repeat Last Edit** menu item, 59
 - RepeatLast** command, 59, 312
 - Replace All** button, 30
 - Replace** button, 30
 - Replace Then Find** button, 30
 - repository, 138
 - ResetTags** command, 70, 308
 - Restored breakpoints overwrite breakpoints set by scripts**
 - configuration option, 261
 - Return** command, 298
 - reuse** command line option, 17
 - Reuse Data Explorer** configuration option, 237
 - Reuse editor windows** configuration option, 252
 - ReverseWord** command, 299
 - Revert** command, 57, 287
 - Revert to Backup** menu item, 57
 - Revert To Date** menu item, 73
 - Revert To History** menu item, 73
 - Revert to Saved** menu item, 57
 - Revert To Version** menu item, 73
 - RevertDate** command, 73, 314
 - RevertHistory** command, 73, 314
 - RevertToBackup** command, 57, 314
 - RevertVersion** command, 73, 314
 - Right** command, 299
 - Right-click mouse action, 312, 338
 - RightArrow key, 299, 325
 - RightD** command, 299
 - RightSome** command, 299
 - size of, 253
 - Root of checkout** configuration option, 216
 - runRcScripts** configuration option, 247
- ## S
- “s” (step) and “n” (next) are blocking by default** configuration option, 235
 - Save arguments between sessions**
 - configuration option, 261
 - Save** command, 56, 287
 - Save command history between sessions**
 - configuration option, 260
 - Save Commit Log** menu item, 153
 - Save Configuration As** menu item, 74

Index

Save Configuration as User Default

menu item, 74

Save data explorers between sessions

configuration option, 260

Save debugger window position

configuration options, 261

Save window positions and sizes

configuration option, 208

SaveAll

command, 287

menu item, 57

SaveAs command, 57, 287

SaveConfig command, 74, 283

SaveConfigToFile command, 74, 283

saveDebugServer configuration

option, 262

saveRunArguments configuration

option, 261

saveViewWindows configuration

option, 260

scientific notation, 52

scratch files, editing, 69, 311

script file

command line, 180

global, 180

program, 180

user, 180

scripts, 179

Scroll bar width in pixels configuration

option (deprecated), 270

scrollBarWidth configuration option

(deprecated), 270

ScrollHLocation configuration option

(deprecated), 270

scrollLocation configuration option

(deprecated), 270

Search

command, 60, 301

dialog box, 28 to 30

keyboard shortcuts, 332

Search in Files, 28, 33 to 34

dialog box, 28, 33

menu item, 68

results window, 34

searching, 28

case-sensitive, 30, 33, 209

functions, 27

in all open files, 28, 33 to 34

incremental, 28

interactive, 28 to 29

tips, 29

with regular expressions, 31 to 32

with Search dialog box, 28 to 29

with Search in Files dialog box, 33

with wildcards, 31

SecondarySelectAll command, 304

SecondarySelectionAdjust

command, 305

SecondarySelectionExtend

command, 305

SecondarySelectionReplace

command, 305

SecondarySelectionReplaceClip

command, 305

SecondarySelectionStart command, 305

SecondarySelectLine command, 304

SecondarySelectWord command, 304

select *color* configuration option, 266

select text keyboard shortcuts, 331

select version control system, 122

SelectAll command, 60, 305

selecting languages, 62

SelectionAdjust command, 305

SelectionDrop command, 284

SelectionExtend command, 305

SelectionGrab command, 305

selectionMarginWidth configuration

option, 258

SelectionStart command, 305

SelectionStartDrag command, 284

SelectionStartDragAdd command, 284

SelectLanguage command, 62, 288

Index

- SelectLine** command, 67, 305
- SelectMatch** command, 305
- SelectRange** command, 306
- SelectToLines** command, 306, 309
- SelectToMatch** command, 63, 306
- SelectWord** command, 306
- serverPollinterval** configuration
 - option, 238
- serverTimeout** configuration option, 238
- Session** tab, 260
- Set INTEGRITY Distribution** menu
 - item, 92
- Set u-velOSity Distribution** menu
 - item, 92
- setBpAtAdrrinitWhenExecing**
 - configuration option, 247
- setting language, 47
- sharedSymbols** configuration option, 247
- shell** command, 247, 311, 321
- shell commands
 - executing, 310
- shellConfirm** configuration option, 247
- Shift+Copy keyboard shortcut
 - (UNIX), 279
- Shift+Cut keyboard shortcut (UNIX), 328
- Shift+DownArrow keyboard
 - shortcut, 302
- Shift+End keyboard shortcut, 332
- Shift+Find keyboard shortcut
 - (UNIX), 333
- Shift+Home keyboard shortcut, 332
- Shift+LeftArrow keyboard shortcut, 331
- Shift+PageUp keyboard shortcut, 331
- Shift+Paste keyboard shortcut
 - (UNIX), 281
- Shift+RightArrow keyboard shortcut, 303
- Shift+UpArrow keyboard shortcut, 331
- Shift+Copy keyboard shortcut
 - (UNIX), 329
- Shift+Cut keyboard shortcut (UNIX), 280
- Shift+[Double Right-click] mouse
 - action, 338
- Shift+DownArrow keyboard
 - shortcut, 331
- Shift+End keyboard shortcut, 303
- Shift+Find keyboard shortcut
 - (UNIX), 301
- Shift+Home keyboard shortcut, 304
- Shift+LeftArrow keyboard shortcut, 303
- Shift+[Left-click] mouse action, 336
- Shift+PageUp keyboard shortcut, 303
- Shift+Paste keyboard shortcut
 - (UNIX), 329
- Shift+[Right-click] mouse action, 338
- Shift+RightArrow keyboard shortcut, 332
- Shift+UpArrow keyboard shortcut, 304
- shortcut menu
 - Editor, 21
- shortcuts, MULTI
 - creating, 113
 - exporting, 114
 - importing, 114
 - overview, 100
 - saving, 114
- Show History**
 - menu item, 72
- Show Last Edit** menu item, 22, 72
- Show locations of variables for “print”**
 - command configuration option, 235
- Show position in non-GUI (-nodisplay)**
 - mode configuration option, 235
- Show tool commands (-commands)**
 - configuration option, 228
- Show tooltips** configuration option, 210
- Show variable values in tooltips**
 - configuration option, 231
- Show version control information on**
 - file chooser dialog box configuration option, 265
- showAddress** configuration option, 235
- ShowContextMenu** command, 312

Index

- showGrepCommand** configuration
 - option, 224
 - ShowHistory** command, 72, 314
 - showhistory** command line option, 17
 - ShowLastEdit** command, 72, 315
 - showPosinNoDisplayMode** configuration
 - option, 235
 - showProgress** configuration option, 226
 - ShowView** command, 315
 - silentlyReloadSymbols** configuration
 - option, 247
 - Single-Thread Build** configuration
 - option, 227
 - SOF** command, 299
 - SOL** command, 299
 - SOL0** command, 300
 - SOL1** command, 300
 - SOLSecondary** command, 306
 - Source Code Font** configuration
 - option, 212
 - Source Files** configuration option, 264
 - sourceFilesDir** configuration option, 264
 - sourceFileSet** configuration option, 264
 - SourceSafe version control
 - automatic checkout, 215
 - configuration, 124, 211
 - database location, 216
 - integration, 132
 - path, 215
 - root of checkout, 216
 - Spaces per indent for Ada** configuration
 - option, 258
 - SpecialTag** command, 308
 - specify
 - alternate editor, 320
 - square brackets [], matching, 63
 - startedConnectionFg** *color* configuration
 - option, 269
 - starting the Editor, 16
 - as stand-alone program, 17
 - from the **Build Details** window, 18
 - from the Debugger, 18
 - from the Launcher, 16
 - from the Project Manager, 18
 - Starts line** check box, 31
 - Starts word** check box, 31
 - Startup** action sequence, 103
 - startup files, 178, 180
 - status bar, 22
 - status box, 22
 - Status** configuration option, 267
 - Stepping over longjumps** configuration
 - option, 238
 - stepToBpignores...** configuration
 - option, 248
 - StopSearch** command, 301
 - string** *color* configuration option, 268
 - Subversion version control
 - configuration, 123, 211
 - integration, 134
 - path, 215
 - synchronous** configuration option, 225
 - syntax color settings configuration
 - option, 268
 - syntax coloring, 49
 - Syntax Coloring** configuration
 - option, 267
 - syntax definition file, 49
 - creating, 47
- ## T
- Tab**
 - command, 294
 - key, 292, 334
 - Tab size** configuration option, 253
 - tabsAreSpaces** configuration option, 252
 - tag files
 - creating, 27
 - reset, 70, 308
 - target connections
 - auto-save, 262

Index

- sharing, 229
 - targetWindowSwitchViewOnBpHit**
 - configuration option, 248
 - Taskbar Organizer
 - configuring, 216
 - menu options, 198
 - overview, 197
 - taskMatchCriteria** configuration option, 239
 - tbTypeBg** configuration option, 249
 - tbTypeFg** configuration option, 249
 - Temp file directory** configuration option, 257
 - text selection keyboard shortcuts, 331
 - third-party tools
 - editors with the MULTI environment, 320
 - integrating the editor, 321
 - version control systems, 320
 - working with MULTI, 320
 - title bars
 - hide, 224
 - naming, 224
 - tog** command, 246
 - Toggle Write Permission**
 - menu item, 57
 - ToggleErrorView** command, 296
 - ToggleReadOnly** command, 63, 288
 - ToggleWritePermission** command, 57, 288
 - toolbar, 20
 - buttons, 20
 - customizing, 183
 - toolCommands** configuration option, 228
 - Tools** menu, 68
 - tooltips**
 - configuration option, 210
 - enabling and disabling, 210
 - show variable values, 231
 - traceAlwaysOn** configuration option (deprecated), 270
 - Translate DWARF debugging information** configuration option, 237
 - Translate stabs debugging information**
 - configuration option, 237
 - transpose characters, 280
 - Tree Browser colors, 249
 - [Triple Left-click] mouse action, 337
 - [Triple Middle-click] mouse action, 337
 - TruncateSearch** command, 301
 - types, color in browser, 249
 - typographical conventions, xii
- ## U
- UnComment** menu item, 22, 66
 - UnCommentBlock** command, 66, 278
 - Undo**
 - button, 30
 - command, 59, 312
 - key (UNIX), 312, 328
 - menu item, 21, 59
 - unifyViewWindows** configuration option, 237
 - Unindent**
 - command, 66, 293
 - menu item, 66
 - unions, color in browser, 249
 - UNIX
 - file chooser dialog box, 76
 - Print Setup** dialog box, 78
 - printing, 78
 - Up** command, 300
 - UpArrow key, 300, 324
 - Update** menu item, 71
 - UpperCase** menu item, 66
 - UpperCaseBlock** command, 66, 278
 - UpSome** command, 300
 - size of, 253
 - usage** command line option, 17
 - Use Color Offsets** configuration option, 231

Index

Use command window configuration
option, 214
Use icons for buttons configuration option
(deprecated), 270
Use lock files (-lockout) configuration
option, 228
Use MULTI Editor on errors
configuration option, 227
Use Preset Colors configuration
option, 231
**Use procedure relative line numbers (vs.
file relative)** configuration option, 230
Use Recent Commit Log menu item, 153
Use XTerm configuration option, 214
useLockFiles configuration option, 228
useLowPriority configuration option, 228
user
configuration file, 176
script file, 180
user configuration file, 176
clearing, 179
override, 176
saving, 172
User Directories configuration
option, 264
user script file, 180
_user_cfg_dir Launcher
variable, 117
UserName command, 294
useWmPositioning configuration
option, 225
Utility Program Launcher, 70, 90, 311

V

variables, *see* MULTI Launcher variables
VCBuffer command, 315
verifyHalt configuration option, 230
versCtrl_ClearCaseAuto... configuration
option, 215
versCtrl_ClearCasePath configuration
option, 215

versCtrl_CustomAutoCheckout
configuration option, 215
versCtrl_CvsPath configuration
option, 215
versCtrl_PvcsAutoCheckout
configuration option, 215
versCtrl_RcsAutoCheckout
configuration option, 215
versCtrl_SourceSafeAuto... configuration
option, 215
versCtrl_SourceSafeDatabase
configuration option, 216
versCtrl_SourceSafePath configuration
option, 215
versCtrl_SourceSafeRoot configuration
option, 216
versCtrl_SubversionPath configuration
option, 215
version control, 122
auto-checkout, 72, 126
auto-detect, 122 to 123, 211
CheckIn, 71
CheckOut, 71
ClearCase, 124, 128, 211
configuration options, 215
CVS, 123, 211
disabling, 123, 211
discard changes, 72
enabling, 72
indicator, 22
location of binary, 215
merging multiple file versions, 70, 311
place file under, 72
PVCS, 124, 130, 211
RCS, 123, 131, 211
Revert to Date, 128
Revert to History, 128
Revert to Version, 128
reverting to backup version, 57
reverting to previous version, 73, 128

Index

- reverting to previously saved
 - version, 57
- reverting to specific date, 73
- reverting to specific version, 73
- show history, 72
- show last edit, 72, 127
- showing information in file
 - chooser, 265
- SourceSafe, 124, 132, 211
- Subversion, 123, 211
- supported systems, 122
- Version Control** configuration option, 211
- Version** menu, 71
- versionControlType** configuration
 - option, 211
- Vertical scroll bar** configuration option
 - (deprecated), 270
- vi Editor
 - command line arguments, 214
 - configuration option, 210
 - path, 214
 - use command window, 214
 - use XTerm, 214
- View** menu, 62
- View unsigned char as integer**
 - configuration option, 231
- viewDef** configuration option, 250
- viewsAreChildrenMode** configuration
 - option, 232
- Visual SourceSafe integration, 132
 - See also* SourceSafe
- VSS (Visual SourceSafe), 132
 - See also* SourceSafe
- W**
- %w special character, 196
- warnOnBpReplacement** configuration
 - option, 251
- warnOnCmdAdrLine...** configuration
 - option, 251
- Warp pointer** configuration option, 210
- WGUtils** command, 70, 311
- wildcards in searches, 31
- window alignment, 199
 - See also* window docking
- Window** configuration option, 220
- window docking
 - application configuration options, 220
 - configuring, 218
 - global configuration options, 218
 - overview, 199
 - UNIX limitations, 200
- window focus, configuring, 201
- window grouping, 197, 199
 - See also* window docking
- window identification number (wid), 196
- window manager
 - enabling, 225
- window organization, 197
- window positioning, 225
- window positions and sizes
 - saving, 208
- Windows** configuration option, 213
- Windows** menu, 74
- Word** command, 300
- Word wrap** configuration option, 258
- working with comments, 38
- workspace action sequences
 - action tree, 97
 - creating, 107
 - deleting, 109
 - editing, 107
 - overview, 100
 - running, 112
 - Startup**, 103
- workspace actions
 - action tree, 97
 - creating, 109
 - editing, 109
 - managing running, 112
 - overview, 100
 - running, 112

Index

workspace drop-down menu, 94
workspace Launcher variables, 116
workspaces
 changing properties of, 105
 creating, 102
 drop-down menu for, 94
 exporting, 106
 importing, 106
 INTEGRITY BSP, creating, 103
 opening, 107
 overview, 5, 100
 saving, 106
Wrap column configuration option, 258
Wrap indent offset configuration
 option, 259

 _ws_file_dir Launcher variable, 117
 _ws_file_name Launcher
 variable, 116
 _ws_file_path Launcher
 variable, 117
 _ws_name Launcher variable, 116
 _ws_working_dir Launcher
 variable, 116

X

X-server, 222
X-window, 225
XORmacs configuration option, 222
XTerm configuration option, 214